FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# Juntando Modelação 3D Manual e Procedimental

**Rui Miguel Barros Gonçalves**


Mestrado Integrado em Engenharia Informática e Computação

Orientador: Rui Rodrigues

Co-orientador: Pedro Silva

Co-orientador: João Jacob

1 de Fevereiro de 2016

# Juntando Modelação 3D Manual e Procedimental

**Rui Miguel Barros Gonçalves**

Mestrado Integrado em Engenharia Informática e Computação

1 de Fevereiro de 2016

# Resumo

Na área da modelação tridimensional existem duas grandes vertentes: modelação manual e modelação procedimental. Na primeira, o utilizador modela o objeto interagindo diretamente com o resultado através de ferramentas virtuais que, em alguns casos, se aproximam daquelas utilizadas por modeladores em objetos reais; desta forma permitindo um controlo direto e fino sobre o que está a ser produzido. Em modelação procedimental, os objetos são construídos de forma automática a partir de várias condições definidas pelo utilizador, através de regras textuais numa linguagem dedicada ou de grafos de fluxo de dados, permitindo um produção rápida em em grandes quantidades de conteúdo 3D. No entanto, nem sempre a interação com estes métodos é a mais intuitiva. Em situações em que é necessário introduzir detalhes particulares, o processo de criação procedimental torna-se demasiado trabalhoso. Para esses casos, a modelação manual seria a mais adequada para efetuar correções ou ajustes em detalhes. É assim importante encontrar um meio que permita combinar as vantagens da modelação manual com as vantagens da modelação procedimental, para que os potenciais utilizadores possam criar modelos tridimensionais com maior eficiência.

Esta dissertação têm como objetivo avaliar possíveis abordagens de integração das duas vertentes de modelação e o impacto da respetiva utilização. Para tal, foi implementada e testada uma solução que visa integrar a ferramenta de modelação procedimental *Construct* na ferramenta de modelação manual *Blender*.

# Abstract

The domain of tridimensional modelling contains two major approaches: manual modelling and procedural modelling. In the first, the user models the object interacting directly with the result through virtual tools which, in some cases, work similarly to those utilized by modelers in real objects; thus allowing for direct and rigorous control over what is being produced. In procedural modelling, the objects are created automatically through several user-defined conditions, by textual rules in a dedicated language or data-flow graphs, allowing for quick and large-scale production of 3D content. On the other hand, interaction with these methods isn't always the most intuitive. In cases where it's necessary to introduce particular details, the procedural generation process becomes too taxing. For those situations, manual modelling would be the most adequate to perform corrections or small adjustments over details. So, it is important to find a medium which permits combining the advantages of manual modelling with those of procedural modelling, so that potential users may create tridimensional models with greater efficiency.

This dissertation has as its objective the evaluation of possible integrations of the two main approaches of 3D modelling and the impact of the corresponding usage. To that purpose, a solution that aims to integrate the procedural modelling tool *Construct* in the manual modelling tool *Blender* was implemented and tested.

# Agradecimentos

Começo por agradecer aos meus orientadores: o Professor Rui Pedro Amaral Rodrigues, Pedro Amorim Brandão da Silva e o João Tiago Neto Jacob, pelo seu trabalho, conselhos e apoio, sem os quais esta dissertação não seria concluída. Agradeço também ao Professor Pedro Cardoso da Faculdade de Belas Artes da Universidade do Porto pelo apoio dado no decorrer desta dissertação.

Agradeço a todos os membros da Faculdade de Engenharia da Universidade do Porto e do Departamento de Engenharia Informática e Computação por me motivarem e ensinarem durante destes anos.

Agradeço em particular ao meu colega e amigo João Almeida pelo apoio que me deu na revisão deste documento. Finalmente, gostaria de agradecer aos meus país, irmão, amigos, colegas de curso, e familiares pelo apoio que me deram ao longo do meu percurso académico.

Rui Miguel Barros Gonçalves

*"What we usually consider as impossible are simply engineering problems... there's no law of physics preventing them."*

Dr. Michio Kaku

# Conteúdo

# Lista de Figuras

# Lista de Tabelas

# Abreviaturas e Símbolos

| | |
|---|---|
| API | Application Programming Interface |
| CAD | Computer-Aided Design |
| CASE | Computer-Aided Software Engineering |
| CGI | Computer-Generated Image |
| JSON | JavaScript Object Notation |
| MP | Modelação Procedimental |
| GP | Geração Procedimental |
| MM | Modelação Manual |
| SUS | System Usability Scale |
| TCP | Transmission Control Protocol |

# Capítulo 1

# Introdução

A criação de ambientes virtuais é um tópico muito relevante em computação gráfica e realidade virtual, tendo aplicações na indústria do entretenimento para a produção filmes e videojogos, entre outros. Na indústria do cinema, estes ambientes virtuais são utilizados para criar imagens impossíveis de reproduzir através de meios práticos. Na indústria dos videojogos, cada vez mais encontramos jogos em ambientes de maior dimensão e com um elevado nível de detalhe (Figura 1.1).

Em modelação, a abordagem tradicional é a de Modelação Manual, que permite ao artista controlo completo e fino do objeto a ser criado. No que toca à criação de cenários de elevadas dimensões, como montanhas, cidades, estradas e vegetação, a abordagem tradicional torna-se cada vez menos viável. A introdução de métodos procedimentais na geração destes conteúdos permite produzir resultados de alta qualidade com alguma facilidade, produzindo cenários numa fração do tempo e do custo correspondentes à respetiva produção por meio de modelação manual.

Apesar do grande impacto positivo que esta abordagem trouxe para a indústria, de um ponto de vista artístico, nem sempre o resultado é o mais desejado; desta forma podendo necessitar da introdução de pormenores e alterações finas à respetiva geometria para produzir um resultado único e mais próximo da visão artística do criador. Para tal, têm sido desenvolvidas técnicas para uma interação mais intuitiva com a modelação procedimental, tentado dar ao modelador um melhor controlo sobre os métodos automatizados que lhes estão subjacentes, não possuindo ainda o nível de controlo fino que uma ferramenta de modelação manual pode oferecer.

Figura 1.1: Captura de ecrã do vídeo jogo *Crysis 3* da *Electronic Arts*

## 1.1 Contexto

Esta dissertação insere-se na área da computação gráfica, mais especificamente na área da modelação tridimensional, geração procedimental de modelos 3D e interação com os mesmos. Estas áreas intersectam-se, em rumo a uma melhor interação com recursos de geração procedimental num ambiente de modelação manual.

Esta dissertação foi realizada na Faculdade de Engenharia da Universidade do Porto no Laboratório de I&D de Computação Gráfica, Interação e Jogos.

## 1.2 Motivação e Problema

Os avanços no desenvolvimento de hardware e software resultam numa necessidade de conteúdo tridimensional maior e mais detalhado, criando uma maior carga nas equipas que os produzem. Como mencionado previamente, a utilização de modelação procedimental para a criação rápida de ambientes tridimensionais é altamente apelativa para a indústria, permitindo poupar tempo e dinheiro comparativamente a uma abordagem manual.

No entanto, esta abordagem dificulta ao modelador a introdução de pormenores, pois os métodos procedimentais são definidos tipicamente por um conjunto de regras e parâmetros, de uma forma não intuitiva ou de controlo limitado. A simples modificação de um parâmetro pode criar uma reação em cadeia de modificações que alteram por completo o modelo gerado. É assim apelativo encontrar um método que melhor aproveite as vantagens das duas abordagens, permitindo aos modeladores uma produção de conteúdo tridimensional mais rápida e controlada.

## 1.3 Objetivos

O objetivo principal deste trabalho é desenvolver uma solução que melhor aproveite as vantagens das duas principais abordagens à modelação 3D; para tal, é possível identificar os seguintes objetivos específicos:

- Investigar o estado da arte da área da modelação procedimental e modelação manual, em particular os avanços realizados na interação com os métodos utilizados em modelação procedimental;
- Estudar as metodologias possíveis para abordar o problema;
- Desenvolver uma solução utilizando uma abordagem possível com os recursos disponíveis;
- Testar a solução com utilizadores proficientes em modelação de conteúdo 3D.

## 1.4 Estrutura do documento

Esta dissertação encontra-se dividida em 6 capítulos, anexos e referências. O capítulo presente consiste numa descrição do problema, da motivação e dos objetivos propostos para a correspondente realização. No capítulo 2 é realizado um levantamento do estado da arte nas áreas da modelação procedimental e modelação manual, em especial os avanços realizados na interação com os métodos utilizados em modelação procedimental. No capítulo 3, são discutidas as metodologias estudadas para a melhor interação entre as duas abordagens à modelação, sendo também apresentada com maior detalhe a metodologia a ser utilizada.

No capítulo 4, são descritos alguns detalhes de implementação da solução desenvolvida baseada na metodologia discutida no capítulo 3.

No capítulo 5, é apresentada a metodologia de teste utilizada para avaliar a solução desenvolvida, sendo também apresentados os resultados obtidos.

No capítulo 6, o final da dissertação, são resumidas algumas constatações e conclusões realizadas sobre esta dissertação. É efetuada também uma listagem de alguns pontos para potencial trabalho futuro.

# Capítulo 2

# Estado da Arte e Trabalhos Relacionados

Neste capítulo será realizada uma revisão bibliográfica que visa apresentar todos os avanços realizados nas áreas da interação com os métodos de geração procedimental. Serão apresentados os tipos de conteúdo passíveis de serem produzidos por meio destes métodos, bem como as técnicas utilizadas para a respetiva interação e geração. Irão também ser apresentadas as abordagens existentes para uma modelação manual e as ferramentas existentes no mercado.

## 2.1 Interação intuitiva para a Modelação Procedimental de mundos virtuais

O paradigma de interação com métodos de geração procedimental é tradicionalmente funcional. Cada procedimento pode ser considerado uma caixa negra de operações que dependendo de certos valores de entrada produz um ou mais resultados. A interação com estes nem sempre é a mais intuitiva pois para atingir o resultado desejado envolve em repetidas tentativa e erro por parte do utilizador. No entanto, em anos recentes tem existido um avanço considerável nesta área, tentado conceder ao utilizador maior controlo sobre os métodos de geração procedimental, com abordagens como *Sketch-based*, editores visuais e Modelação Procedimental Invertida.

### 2.1.1 Técnicas *Sketch-based*

Esta abordagem tenta aproximar o tipo de interação com que os principais utilizadores de métodos de Modelação Procedimental estão familiarizados. Noções como *Brushes* e *Sketches* são alguns dos conceitos mais populares entre *designers* profissionais.

#### 2.1.1.1 Terrenos

De Carpentier et al. [dCB09] propuseram um método de edição de terrenos no qual é dada ao utilizador a possibilidade de alterar o *heightfield* através de *procedural brushes*, tentando combinar a interação manual localizada com a modificação da parametrização por completo de *heightfields*. Esta abordagem é capaz de produzir alterações a tempo real, graças à respetiva implementação com *shaders*.

Smelik et al. [STdKB10], seguindo a respetiva proposta de modelação de mundos virtuais [STdKB08], propõem a utilização da abordagem *Procedural sketching* e demonstram-na num protótipo intitulado de *SketchaWorld*. Esta abordagem é dividida em dois principais modos de interação (Figura 2.1): *Landscape Mode* e *Feature Mode*.

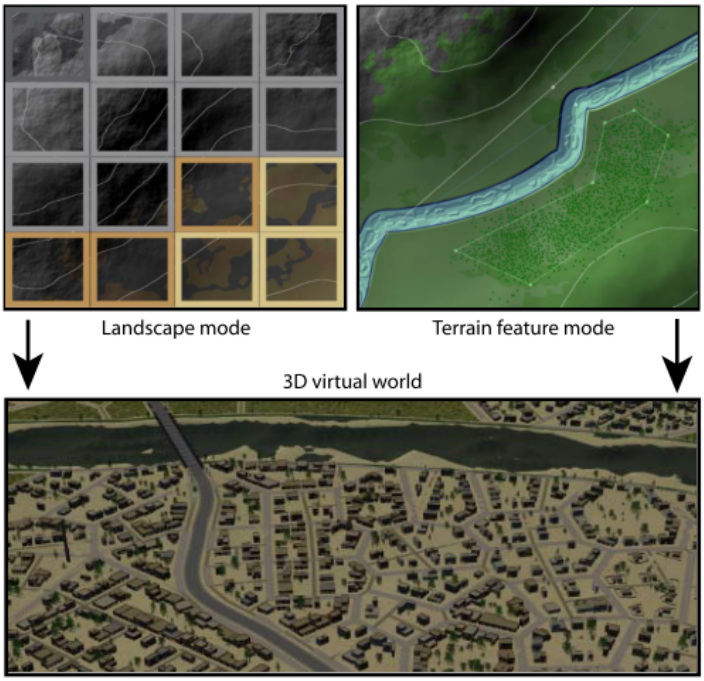


Figura 2.1: *Procedural sketching of virtual worlds* por Smelik et al. [STdKB10]

No primeiro, o utilizador interage com uma vista *top-down* do terreno, divido em grelha e é capaz de definir a informação geológica de cada célula através do uso de *Brushes*. Em *Feature Mode*, o utilizador manipula pontos de controlo, correspondendo a vetores ou polígonos, que delimitam a localização de cidades, rios e vegetação. A ferramenta fornece *feedback* da construção do terreno a um ritmo considerável para uma interação fluida. Esta também dispõe de funcionalidades como *undo* e *redo* para uma melhorar a interação entre o utilizador e a ferramenta, permitindo assim a introdução de alterações e, caso estas não sejam desejadas pelo utilizador estas podem ser desfeitas rapidamente. Gain et al. [GMSe09] propõem uma solução que permite ao utilizador esboçar as silhuetas e limites de corpos montanhosos a serem gerados.

#### 2.1.1.2 Estradas

Num esforço de melhorar a interação com *path layouts*, Singh et al. [SM08] desenvolveram a ferramenta *Drive*, que permite a conceptualização de redes rodoviárias tridimensionais por meio de esboços, que são posteriormente aproximados a clotóides (Curvas em que o raio varia linearmente). Com esta abordagem é possível produzir itinerários em que as transições entre raios de curvatura são lineares, tornando-os assim adequados para a condução. A ferramenta também introduz *Terrain sensitive sketching* que, por meios automáticos, coloca sinalização no percurso e vegetação nas redondezas.

#### 2.1.1.3 Vegetação

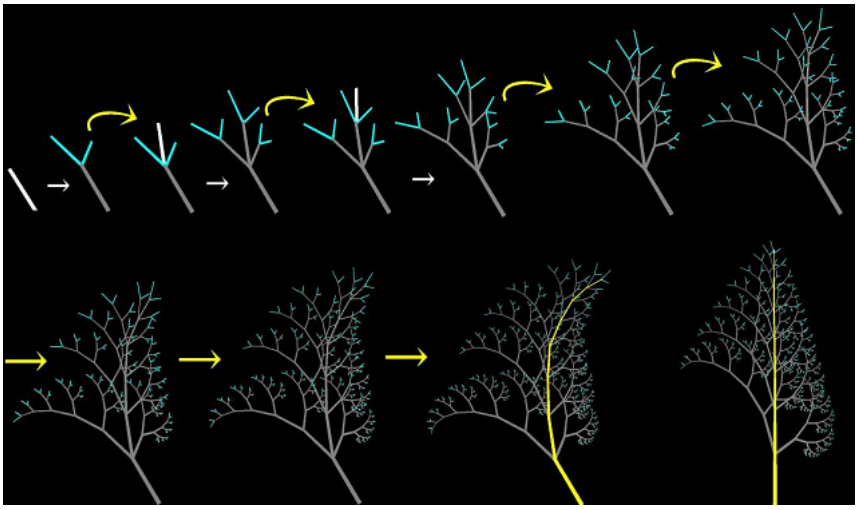

Figura 2.2: Crescimento de vegetação seguindo um eixo esboçado pelo utilizador, por Ijiri et al. [IOI06b, IOI06a]

O trabalho de Ijiri et al. [IOI06b, IOI06a] apresenta uma técnica que, através de traçados definidos pelo utilizador (Figura 2.2), permite a manipulação da direção de crescimento de vegetação produzida por *L-Systems*, uma abordagem já muito estudada para a criação de vegetação, proposta por Lindenmayer [Lin68] como um tipo de gramática formal capaz de representar padrões e geometria fractal existente na vegetação. Em 1988 foi proposto por Prusinkiewicz [PLH88] a utilização de *L-systems* para a modelação de vegetação. Ao longo das ultimas décadas, têm sido realizados avanços consideráveis na produção de vegetação realística [WP95] e a respetiva interação com o seu meio ambiente [MP96, Pru00], tentado assim produzir vegetação que melhor simule o comportamento físico e biológico de vegetação real.

Mais recentemente, o trabalho de Palubicky et al. [PHL+09] foi expandido e transportado para uma aplicação para dispositivos *iPad* em que o modelador tem um controlo fino sobre o crescimento de uma árvore. Utilizando o conceito de *tree self-organization* Longay et al. criaram

a ferramenta *TreeSketch* 2.3 que estimula a competição entre os ramos de uma árvore por espaço e luz dentro dos parâmetros e *inputs* visuais dados pelo utilizador [LRBP12].

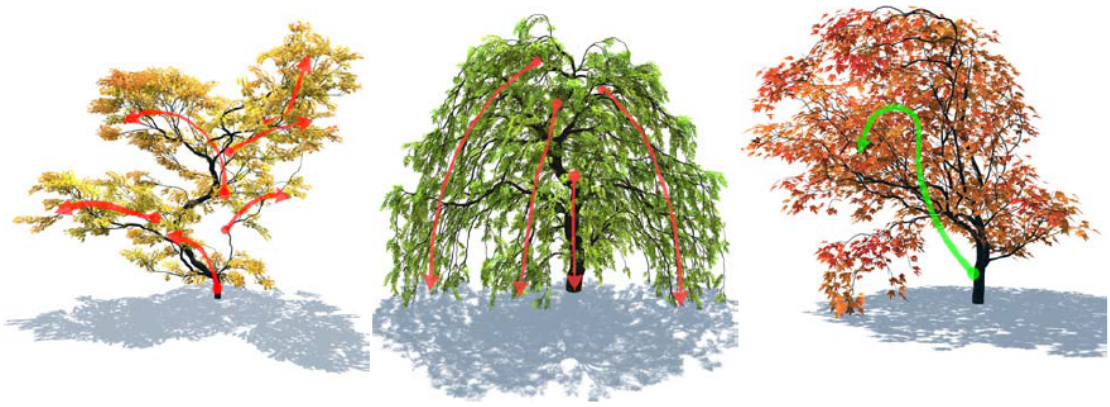

Figura 2.3: Exemplos de árvores geradas com respetivos movimentos na aplicação *Tree Sketch*

### 2.1.2 Editores Visuais

Numa tentativa de melhorar a interação com modelação procedimental baseada em gramáticas [WWSR03, MWH+06], Lipp et al. propuseram uma abordagem que permite a manipulação direta do resultado produzido por uma gramática. O utilizador pode manipular diretamente no objeto as respetivas dimensões e configurações. [LWW08].

Mais recentemente, Krecklau et al. [KK12] introduziram uma abordagem que permite a geração de cenas a partir de primitivas de alto nível e a correspondente manipulação por interação direta na cena; consequentemente permitindo a utilizadores menos experientes a utilização de gramáticas procedimentais complexas, sem a necessidade de interação textual com as mesmas.

#### 2.1.2.1 *Visual Node Based*

Editores *Visual Node Based* são utilizados em muitas aplicações comercias como meio de dar ao utilizador *feedback* visual do fluxo de informação. Encontra-se em editores de *shaders* [Hol15, Aut15b], editores de animações [Tec15] e em editores de texturas [Cod15]. A vasta utilização da técnica na área de produção de conteúdo 3D por utilizadores com pouca experiência de programação levou a que investigadores a utilizassem para a especificação de gramáticas de produção.

Nesse contexto, Patow [Pat12] desenvolveu uma solução para a construção e manipulação de gramáticas por meio de um editor visual, onde nós, que representam operações geométricas dos correspondentes antecessores, transmitem o resultado aos nós sucessores. Na mesma publicação, Patow realça a importância que esta abordagem acarreta para a modelação procedimental de edifícios, e admite que existe ainda muito potencial por explorar.

Mais recentemente, o trabalho de Silva et al. [SMBC13] apresenta uma metodologia de edição em que grafos podem ser encapsulados em nós designados *Component*, capazes de representar construções complexas. Os nós podem ser facilmente introduzidos noutros grafos e editados por meio de *filtros de fluxo*, colocados nas arestas ou por restrições que auxiliam a especificar a utilização dos nós (Figura 2.4).

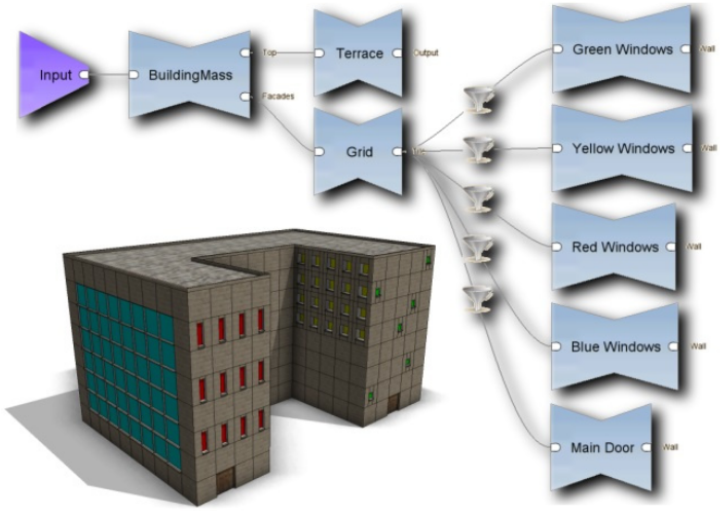


Figura 2.4: Edifício e respetivo grafo de componentes com filtros de fluxo [SMBC13].

### 2.1.3 Modelação Procedimental Invertida

*Inverse PM* é uma técnica em que um dado modelo de entrada é aproximado a uma representação procedimental.

Aliaga et al. [ARB07] apresenta a reconstrução de edifícios por meio de fotografias. Nesta abordagem, o utilizador define os limites das fachadas do edifício criando uma representação minimalista. De seguida é automaticamente aproximada uma gramática que melhor representa o edifício em questão. Com os trabalhos de Š'tava et al. [ŠBM+10] na parametrização de arte vetorial 2D e de Mitra et al. [MGP06] em encontrar simetrias em objetos tridimensionais, Bokeloh et al. [BWS10] aplica estes conceitos para a descoberta de simetrias nos objetos e a consequente aproximação de gramáticas que definam o objeto em questão.

As abordagens atuais focam-se em Modelação invertida de ambientes urbanos, no entanto Vanegas et al. [VGDA+12] demonstram uma metodologia de aproximação de gramáticas capazes de representar, para além de edifícios, sistemas complexos de vegetação aproximados a um objeto já existente.

## 2.2 Modelação 3D Manual

Para a produção de objetos tridimensionais, os modeladores utilizam ferramentas dedicadas que lhes dão um controlo total sobre o modelo a ser construído. Estes variam em metáforas de interface e em público alvo. Nesta secção vão ser abordadas as técnicas de modelação atualmente utilizadas por estas ferramentas e serão listadas as ferramentas existentes no mercado e as respetivas funcionalidades principais.

### 2.2.1 Técnicas de Modelação Manual

#### 2.2.1.1 *Polygon Modelling*

*Polygon Modelling* é uma abordagem para a representação de superfícies tridimensionais utilizando polígonos: estes são representados por vértices e arestas e em grupos formam uma *Polygon Mesh*, que pode ser armazenada de múltiplas formas [TM06, Bau72, Smi06]. É possível criar malhas vértice a vértice, manualmente. No entanto, é comum a respetiva manipulação em software gráfico, concedendo aos modeladores ferramentas interativas como *Blender*, *Autodesk Maya* e *3D Studio Max*.

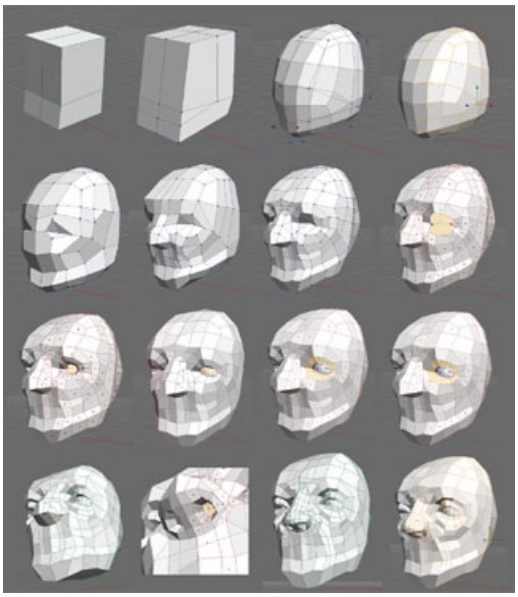

Figura 2.5: Exemplo de modelação utilizando a técnica de *Box Modelling*, [sub14].

Uma das abordagens mais comuns é denominada *Box Modelling*, na qual uma forma geométrica considerada uma primitiva (i.e. cubo, esfera, cilindro) é esculpida iterativamente por passos de subdivisão do objeto e subsequente manipulação da geometria até se obter o objeto final [Jon02, sub14]. A figura 2.5 demonstra da técnica *Box Modelling* para a modelação de uma face. Esta abordagem pode ser considerada uma abordagem Subtrativa, pois é através da redução de um objeto base que é obtido o objeto desejado.

O *polygon modelling* caracteriza-se pelo controlo direto sobre a colocação de cada vértice de uma superfície, permitindo dessa forma um controlo fino e preciso sobre o resultado a produzir.

#### 2.2.1.2 *NURBS*

*NURBS* ou *Non-uniform rational B-spline* é uma generalização de *B-Splines* e Curvas de *Bézier* [KR79, Cab92], um modelo matemático usado para representar curvas e superfícies com grande precisão e flexibilidade (Figura 2.6). É principalmente utilizado em áreas da computação gráfica (CAD, CAM, CAE) onde a precisão dos modelos obtidos é muito importante. Inicialmente estudada por Matemáticos como I. J. Schoenberg [Cob84], foi aplicada em sistemas CAD para a modelação na indústria Automóvel [Bé83, Béz81, Bez68].

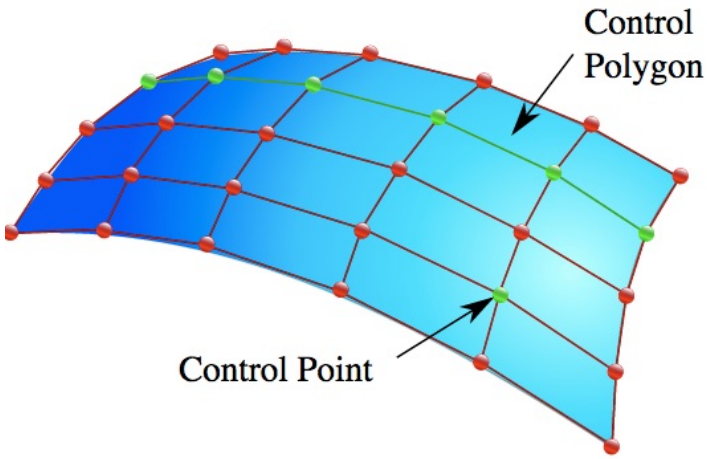


Figura 2.6: Exemplo de superfície com respetivos pontos de controlo criada usada o modelo *NURBS*

#### 2.2.1.3 *Digital Sculpting*

*Digital Sculpting* é a adaptação de técnicas utilizadas para moldar materiais como o barro à modelação 3D. Estas ferramentas permitem ao modelador "empurrar", "puxar", "polir", "vincar"e adicionar ou remover volume. Perry et al. [PF01] introduziram *Kizamu*, um dos primeiros passos na área do *Digital Sculpting*. Os modelos são representados no que é chamado de *Adaptively Sampled Distance Fields* (ADF) [FPRJ00], que permite representar dados volumétricos que tornam o processo de *Digital Sculpting* possível.

### 2.2.2 Ferramentas de Modelação 3D

#### 2.2.2.1 Blender

O *Blender* [ble15a] é uma ferramenta profissional de modelação de código aberto desenvolvida pela Blender Foundation. Suporta uma grande variedade de primitivas geométricas, in-

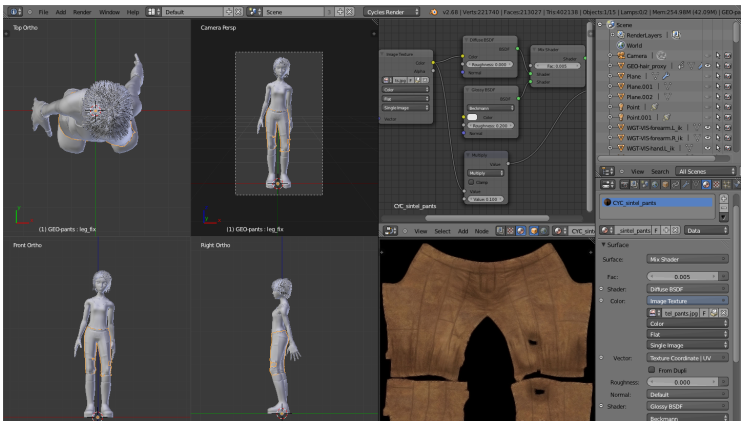

Figura 2.7: Interface da ferramenta *Blender*.

cluíndo *Polygon Meshes*, curvas de *Bezier*, superfícies *NURBS*, *Metaballs* e *Digital Sculpting*. Esta destaca-se por ser gratuita e pela facilidade de criar extensões [ble15b] usando a API *Python* existente [ble15c], que permite adicionar novas funcionalidades à ferramenta.

#### 2.2.2.2 Autodesk Maya

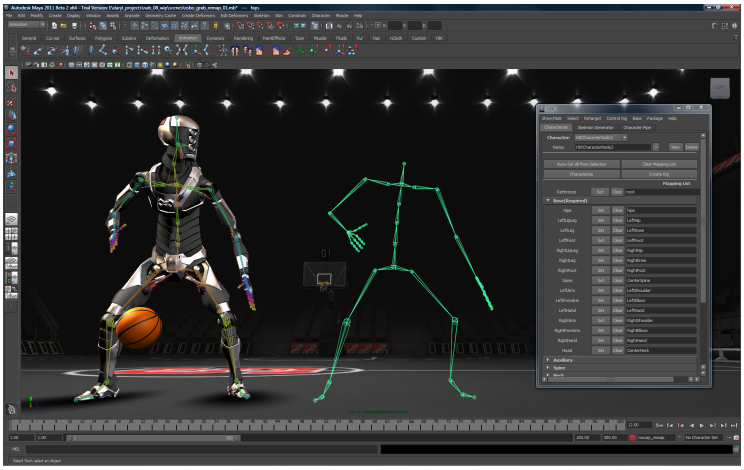

Figura 2.8: Interface da ferramenta *Autodesk Maya*.

O *Autodesk Maya* [Aut15b] é uma ferramenta de produção de conteúdo 3D utilizada nas indústrias do cinema, televisão, videojogos e arquitetura. É de código fechado e a respetiva utilização requer que o utilizador possua uma licença. Tal como o *Blender*, o *Maya* dá aos modeladores ferramentas para estes criarem objetos de uma forma interativa 2.8. Apesar de ser uma ferramenta versátil e capaz de satisfazer as necessidades de todas indústrias acima referidas, o *Autodesk Maya* tende a ser mais utilizado pela indústria do Cinema devido à vasta customização possível de aspetos importantes para esta indústria como Animação e VFX [Aut15c].

#### 2.2.2.3 Autodesk 3D Studio Max

O *3D Studio Max* [Aut15a] é outra ferramenta da Autodesk, em muitos aspetos semelhante ao *Maya*. No entanto, o *3DS Max* está apenas disponível para máquinas de sistema operativo Windows e sempre foi mais utilizado pela indústria dos videojogos para a criação de modelos 3D. Tal como o *Autodesk Maya* este dispõe de poderosos motores de renderização *Ray Tracing*.

#### 2.2.2.4 *Zbrush*

O *Zbrush* [Pix15] é uma ferramenta de modelação 3D baseada nos conceitos de *Digital Modelling*. Dá ao artista múltiplas *Brushes* que permitem manipular o objeto a ser definido com paradigmas transportados de outras áreas artísticas. Pode tornar mais acessível, para um utilizador pouco experiente com alguns paradigmas da Computação Gráfica, a modelação de objetos com considerável detalhe e liberdade artística [Spe11].

## 2.3 Sumário

Após o levantamento do estado da arte, é possível constatar que as ferramentas de modelação manual são a escolha de eleição para contextos em que o modelador deseja o maior nível controlo possível sobre o objeto a ser produzido. No entanto, considerando ferramentas de modelação procedimental, tem existido um esforço por parte desta comunidade para tornar a correspondente experiência de utilização mais acessível a utilizadores pouco familiarizados com o funcionamento do mesmo. Por enquanto, a utilização deste tipo de ferramentas ainda requer que o conteúdo seja criado em ferramentas dedicadas, completamente separadas das ferramentas tradicionais (i.e. *Maya*, *Blender*) essenciais para o trabalho artístico, levando a quebras no fluxo de trabalho.

# Capítulo 3

# Metodologia

Com os desenvolvimentos atuais na área de modelação procedimental apresentados no capítulo 2, é possível verificar que as soluções de MP existentes permitem produzir variados tipos de conteúdo, tais como terrenos, vegetação, edifícios, estradas e cidades; por outro lado, as respetivas integrações em ambientes de modelação manual são algo limitadas, ou apenas disponíveis em ferramentas externas especializadas num único tipo de conteúdo.

Da mesma forma como verificado em 2.1, foram efetuadas várias tentativas pela comunidade envolvida em geração procedimental no sentido a melhorar a interação com ferramentas dedicadas à geração procedimental; no entanto, existe ainda uma grande margem para avanço de uma integração genérica destes processos numa ferramenta de modelação manual. Devido à falta de soluções que se integrem de forma transparente e acessível em ambientes de modelação manual, pretende-se apresentar uma solução com o objetivo de interligar as duas abordagens neste capítulo. Inicialmente será apresentada em pormenor a arquitetura da solução, de seguida o protocolo de comunicação entre os dois ambientes de modelação, e finalmente será proposto um método de trabalho no qual será descrita a interação ideal entre o modelador e a solução proposta.

## 3.1 Método de Trabalho

Numa interação ideal, o modelador teria controlo absoluto sobre os métodos de geração procedimental e seria possível modelar procedimentalmente sobre alterações manuais realizadas sobre o objeto. No entanto, nem sempre é desejado esse nível de controlo e interligação entre as abordagens. Imaginando a modelação de um edifício: este é construído segundo métodos procedimentais, e recebe como entrada um polígono que delimita a fundação, assim como parâmetros numéricos e booleanos que definem a geometria resultante. Depois de modelado procedimentalmente, o objeto é adicionado ao ambiente de modelação manual, onde o modelador pode realizar alterações em detalhes na geometria ou mesmo alterações profundas na geometria do mesmo.

Considerando uma situação em que o modelador realizou alterações profundas na geometria produzida, como, por exemplo, a repartição de uma das fachadas do edifício em três fachadas. Assim que realizadas estas alterações, o utilizador pretende alterar a altura do edifício, alterando o parâmetro correspondente e requisitando uma atualização. Para ser possível manter as alterações realizadas manualmente e, em simultâneo, respeitar as regras definidas pela geração procedimental, seria necessário converter estas alterações em regras possíveis de definir no escopo do ambiente de geração procedimental.

De qualquer das formas, as alterações realizadas no ponto anterior podem simplesmente não fazer sentido após a aplicação sobre o objeto que foi reproduzido seguindo as regras definias por um procedimento. Devido a esta limitação, é assumido que assim que o utilizador requisita uma nova execução de um procedimento, o qual não tem problemas em descartar as alterações executadas manualmente sobre o objeto em questão.

Tendo em consideração esta limitação, é possível estruturar um fluxo de trabalho que melhor explore as vantagens e limitações de ambas as abordagens de geração de conteúdo tridimensional. O utilizador tem uma panóplia de conteúdos procedimentais ao seu dispor que podem ser definidas via parâmetros numéricos e geométricos. O conteúdo é adicionado ao ambiente de modelação manual e os parâmetros podem ainda ser editados alterados ao custo de que as modificações manuais realizadas sobre o objeto sejam descartadas.

## 3.2 Metodologias de Interação com o Utilizador

Em relação à interação com o utilizador, foram estudadas três abordagens para a interação com métodos de geração procedimental em ambiente de modelação manual:

### 3.2.1 Modificadores

O conceito de modificadores é já utilizado em ambientes de modelação manual, e.g. no Blender [Bap14, Ble15e], onde são utilizados para realizar operações sobre objetos na cena sem destruir a geometria subjacente: apenas modificam a forma como os objetos são percecionados pelo motor de renderização. Esta abordagem é útil em situações onde o procedimento a ser executado envolve modificações sobre uma geometria base; como tal, permitindo realizar múltiplas alterações sobre um objeto, podendo estas ser facilmente canceladas, recuperando o objeto original. No entanto, nem todos os recursos de geração procedimental incidem sobre modificação de geometria. Os meios de geração procedimental, em muitas situações, produzem objetos e cenários únicos e independentes de qualquer geometria de entrada.

### 3.2.2 Grafos

A abordagem de interação via grafos é atualmente utilizada num ambiente de modelação manual para a edição de materiais [Ble15d]. Também chamada de *Flow-Based Programming*, esta

abordagem permite apresentar ao utilizador os procedimentos como uma "caixa negra", com possíveis entradas e saídas representadas como ligações de fluxo. A correspondente utilização, na interação com geração procedimental, pode permitir o encadeamento de múltiplos procedimentos. No entanto, a abordagem pode ser mais trabalhosa para o utilizador, introduzindo um nível de interação adicional para cada objeto introduzido na cena.

### 3.2.3 Funções

Esta abordagem disponibiliza ao utilizador cada recurso de geração procedimental como uma função disponível a qualquer momento na interface da ferramenta de modelação manual. Aquando da respetiva invocação, são requisitados aos utilizadores os parâmetros necessários à execução. A abordagem em causa é potencialmente a mais fácil de adotar, por parte dos utilizadores de ferramentas de modelação manual. Devido à simplicidade na interação, assim como a reduzida necessidade de tempo e recursos para a implementação, esta foi a abordagem de interação utilizada na solução desenvolvida.

## 3.3 Arquitetura da Solução

Para abordar este problema, é proposta a utilização de duas aplicações já presentes no mercado: Uma aplicação dedicada à modelação procedimental e uma aplicação de modelação manual. Foi escolhida esta abordagem na medida em que é possível utilizar duas aplicações já existentes, sendo possível dessa forma aproveitar os pontos fortes de cada solução com o intuito de otimizar o processo de criação de novos cenários e conteúdo tridimensional. Esta abordagem pode ser representada como uma solução estruturada numa arquitetura Cliente-Servidor, em que a aplicação de geração procedimental é capaz de fornecer serviços de geração de conteúdo a um ou mais clientes de modelação. Esta abordagem interliga as duas aplicações através de um protocolo de comunicação. Este permite ao utilizador da ferramenta de modelação manual requisitar a produção de conteúdo, seguindo os valores impostos pelo utilizador nos parâmetros do procedimento requisitado. O protocolo é descrito em maior detalhe na secção 3.4.

### 3.3.1 Vantagens

**Fácil integração em diferentes plataformas** Devido à arquitetura ser baseada na comunicação entre duas aplicações por um protocolo de comunicação genérico, é com alguma facilidade que se realiza a interligação com diferentes soluções de geração procedimental ou de modelação manual. Sendo assim possível a expansão para novos ambientes de modelação manual, ou mesmo introduzir novas soluções de geração procedimental.

**Escalabilidade** A arquitetura proposta poderá permitir que esta seja implementada de modo a que um servidor central de geração procedimental sirva, por exemplo, num ambiente empresarial, como um repositório de recursos de geração de conteúdos; desta forma, permitindo

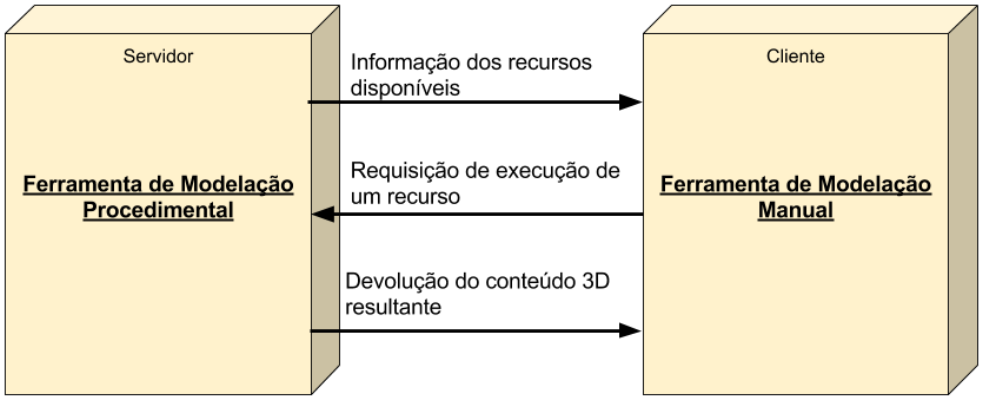


Figura 3.1: Diagrama do sistema proposto

adicionar novo conteúdo assim que necessário por cada modelador ao ambiente de modelação em que estão a produzir conteúdo tridimensional.

### 3.3.2 Desvantagens

**Falta de transparência** A abordagem utilizada isola o funcionamento interno dos métodos de geração procedimental do utilizador. Este apenas sabe o que é possível adicionar à cena, e os parâmetros e geometrias de entrada. Para um utilizador proficiente em geração procedimental, torna-se mais apelativo uma integração mais transparente e que lhe garanta um maior controlo da geometria resultante. Por exemplo, edição dos procedimentos através da interface de modelação manual.

**Latência da comunicação** Esta abordagem pode, em certas situações introduzir latências na interação com o utilizador, pois esta está dependente de comunicações entre duas aplicações completamente independentes.

**Conhecimentos das duas aplicações** Em situações em que é desejado por um utilizador a alteração do funcionamento de um procedimento esta tem de ser realizada fora da aplicação de modelação manual. Pois a solução proposta abstrai-se de um editor de recursos procedimental e trata cada procedimento como uma caixa negra em que dadas umas certas entradas um resultado é obtido e adicionado ao cenário na aplicação de modelação manual.

## 3.4 Protocolo de Comunicação

No que respeita à comunicação entre as duas aplicações, é preferível um protocolo de comunicação legível e extensível, de modo a facilitar a implementação de novas funcionalidades entre as aplicações. Os tipos de mensagens existentes são:

**Requisição de recursos disponíveis** Esta mensagem é a primeira a ser realizada assim que a aplicação cliente inicializa, requisitando ao servidor quais os recursos de modelação procedimental disponíveis. É enviada da aplicação cliente para o servidor, permitindo que este carregue todos os recursos necessários para a geração de conteúdo e que informe a aplicação cliente dos procedimentos disponíveis.

**Informação dos Recursos disponíveis** Esta é a mensagem de resposta a um pedido de informação inicial. A mensagem encapsula toda a informação relativa aos recursos de geração procedimental existentes, para que a aplicação cliente seja capaz de gerar uma interface correspondente e que informe o utilizador de quais os tipos de conteúdo disponíveis para requisição. É também incluída toda a informação relativa aos parâmetros que cada procedimento pode receber.

**Gerar conteúdo** Esta mensagem é enviada pelo cliente e informa o servidor do procedimento disponibilizado pelo servidor a correr e de quais os valores para os parâmetros e entradas. É transmitida informação de qual o procedimento a executar, quais os seus parâmetros de execução e, caso aplicável, a geometria a ser utilizada como base para a execução das operações procedimentais.

**Conteúdo Gerado** Esta mensagem é utilizada pelo servidor como confirmação que o conteúdo requisitado foi produzido e se encontra disponível para ser utilizado pela aplicação cliente.

**Mensagem de erro** Caso ocorra algum erro no servidor, este pode comunicar à aplicação cliente via uma mensagem de erro, que é apresentada ao utilizador pela interface da aplicação cliente.

Este protocolo assume que os recursos de geração procedimental estão organizados em projetos que englobam um conjunto de recursos procedimentais, podendo ser requisitados individualmente e que requerem até dois principais tipos de informação para a correspondente execução:

**Parâmetros** Atribuem valores a variáveis internas ao procedimento, permitindo assim ao utilizador alterar o conteúdo produzido. Cada procedimento pode requisitar um número arbitrário de parâmetros de entrada, os quais podem tomar valores alfanuméricos ou booleanos (Verdadeiro ou Falso).

**Geometria de entrada** Definem uma geometria base, sobre a qual certas operações do procedimento serão realizadas de modo a produzir uma geometria resultante. Por exemplo, no caso da geração de um edifício, o seu perímetro da base pode ser definido por uma geometria.

Esta base servirá de fundação para as operações de geração de um edifício enquadrado na mesma.

# Capítulo 4

# Implementação

Neste capítulo serão descritos em detalhe os passos realizados para a implementação da solução proposta, segundo a metodologia apresentada no capítulo anterior.

Para implementar a metodologia previamente definida, foram selecionadas tecnologias já existentes que satisfizessem as necessidades de uma ferramenta de modelação manual possível de ser servida de funcionalidades de geração procedimental de uma aplicação externa. É necessário que as duas ferramentas sejam capazes de interagir entre si mediante um protocolo de comunicação que permita a troca de mensagens padronizadas entre as duas ferramentas.

Como Aplicação cliente, foi utilizado o *Blender*; como servidor de recursos de geração procedimental, o *Construct*; e para permitir a comunicação entre estas duas ferramentas, foi definido um protocolo em JSON que é utilizado para a comunicação mútua via *Sockets*.

## 4.1 Aplicação Cliente - Ferramenta de Modelação Manual

Na atualidade, existem diversas ferramentas de modelação manual disponíveis no mercado, estas foram listadas no estado da arte; no entanto, é de destacar o *Blender*: uma ferramenta gratuita e open-source de computação gráfica. Esta é capaz de produzir imagens geradas por computador (CGI) de geometria modelada na ferramenta. Este software foi escolhido como a aplicação cliente devido às seguintes vantagens face à respetiva concorrência:

**Gratuito e Open Source** Ao contrário das principais ferramentas concorrentes, o *Blender* é uma aplicação gratuita e de código livre.

**Comunidade vasta e ativa** Comparativamente à concorrência, o *Blender* possui uma comunidade de utilizadores extremamente ativa em fóruns e páginas de suporte. Depois de compilada a tabela seguinte 4.1, em que é possível verificar as estatísticas dos principais fóruns de utilizadores de *Blender*, *3Dsmax* e *Maya*, comprova-se que a comunidade *Blender* é a maior e mais ativa entre as aplicações concorrentes.

Tabela 4.1: Estatísticas dos principais fóruns

| Forum | Utilizadores | Tópicos | Mensagens |
|---|---|---|---|
| Blenderartists.org | 228.649 | 356.260 | 2.936.935 |
| blender.org/forum | 117.957 | 22.856 | 98.144 |
| forums.autodesk.com/t5/maya | desconhecido | desconhecido | 78.656 |
| simplymaya.com/forum | 238.228 | 37.980 | 335.735 |
| forums.autodesk.com/t5/3ds-max | desconhecido | desconhecido | 195.129 |

**API Python** O *Blender* disponibiliza uma *API Python* que permite criar novos elementos na interface e novas ferramentas através de *scripts* em *Python*. Estes *scripts* podem ser configurados para serem inicializados assim que o *Blender* arranca, funcionando assim como *add-ons* ao sistema base.

## 4.2 Aplicação Servidor - Ferramenta de modelação procedimental

Como ferramenta de modelação procedimental, foi utilizado como base o *Construct*, desenvolvida na FEUP por Pedro Silva no âmbito da respetiva dissertação de Doutoramento. O *Construct* é um sistema de geração procedimental genérico que tem a capacidade de produzir uma panóplia de conteúdo de forma automatizada seguindo regras (figura 4.1). As regras são definidas numa linguagem visual de alto nível em grafos de fluxo que definem regras e operações a aplicar de modo a produzir a geometria resultante.

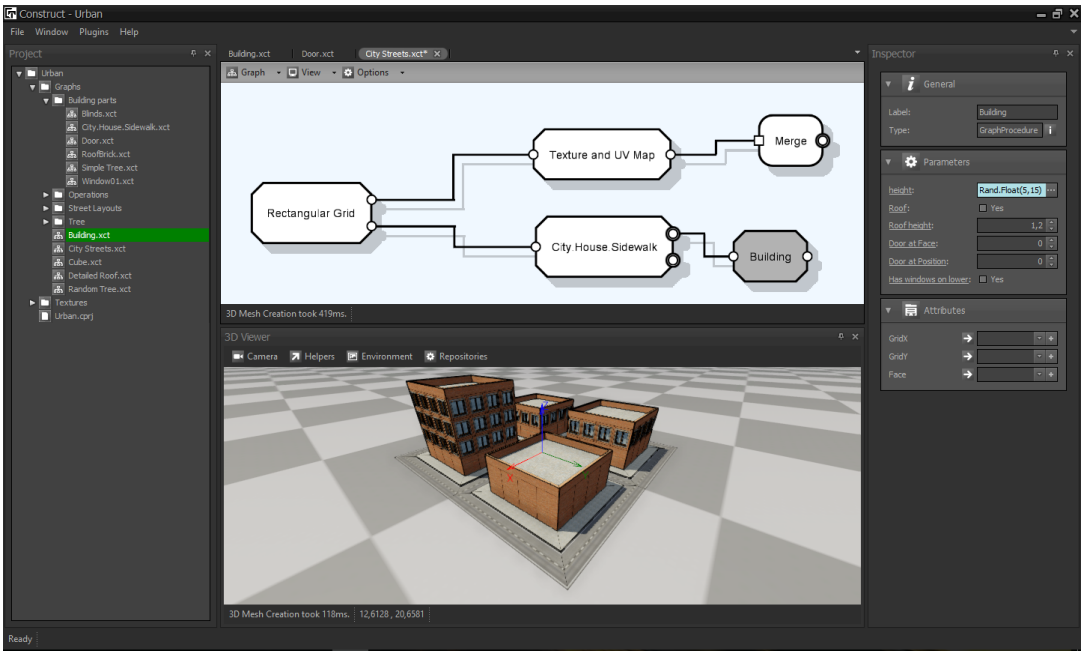


Figura 4.1: Editor *Construct* em execução

O utilizador, através da manipulação de nós básicos que criam e manipulam conteúdo, constrói novos grafos para atingir o resultado desejado. No seguinte grafo de exemplo (figura 4.2), é definido um retângulo de 3 unidades por 2 unidades de dimensão e é aplicada a operação de *extrude* de 2 unidades em que depois de uma execução é possível visualizar que o objeto resultante é um paralelepípedo com as dimensões de 3 por 2 por 2.

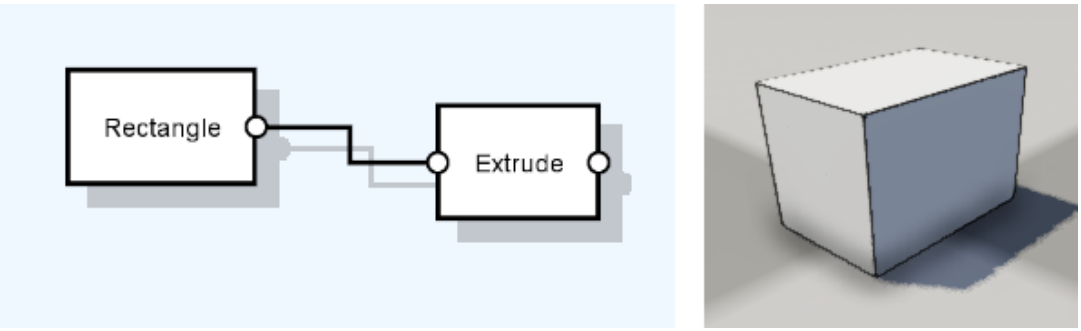


Figura 4.2: Grafo exemplo e respetiva geometria produzida

Para a solução desenvolvida foi utilizada a biblioteca *.NET C#* do *Construct*. Foi implementado um servidor que responde a pedidos de execução de grafos transmitidos via *Sockets TCP*. O resultado da execução de um grafo é codificado como um ficheiro *Wavefront OBJ* e é colocado numa pasta temporária partilhada entre as duas aplicações.

## 4.3 Comunicação

A comunicação entre as duas aplicações necessita de ser capaz de transmitir informação entre ambas, através de um protocolo comum e interpretável pelas duas aplicações distintas. Para tal, foram estudadas as alternativas existentes: *Named Pipes*, *Network Sockets*, *XMLRPC* e *SOAP* foram as tecnologias encontradas que permitem a comunicação entre um *Add-on Blender* em *Python* e uma aplicação *.NET* com acesso aos recursos do *Construct*. No entanto, devido ao foco desta dissertação não ser a comunicação entre as duas aplicações, mas a integração de paradigmas de modelação procedimental num ambiente de modelação manual foi escolhida uma comunicação entre as duas aplicações via *Network Sockets*. Trata-se de uma tecnologia que está disponível em essencialmente qualquer sistema, que seria a mais rápida de implementar e, simultaneamente, que suporta todas as necessidades do sistema. As mensagens são estruturadas num objeto *JSON* que possui sempre como base os campos *Subject* e "Data". Sendo que o campo *Subject* identifica o tipo de mensagem que está a ser enviada e o campo *Data* contem a informação necessária para realizar dita mensagem.

O servidor, após a inicialização, espera que seja requisitada a abertura de um novo projeto Construct; o que é realizado através do pedido *OpenProj*, onde o *add-on* informa sobre o caminho absoluto para a localização do ficheiro do tipo ".cprj". Depois de verificar que o ficheiro em questão especifica um projeto de Construct, o servidor carrega o projeto e, de seguida, é percorrida

```
{
  "Subject": "OpenProj",
  "Data": "C:\\Users\\ruinf\\Documents\\Urban\\Urban.cprj"
}
```

Figura 4.3: Exemplo de uma mensagem JSON utilizada para a abertura de um Projeto Construct

toda a informação relativa aos grafos; com essa informação, é construido um objeto JSON. O objeto JSON acarreta toda a informação necessária para construir a interface necessária, incluindo nomes dos grafos e os respetivos parâmetros, com tipos e entradas de geometria. Na figura 4.4 é possível vermos um exemplo de informação relativa a um projeto Construct que contém apenas um grafo. O grafo em questão gera um edifício e aceita dois parâmetros: *Height*, a altura do edifico; e *Roof Height* a altura do telhado. Este grafo também requer um objeto a ser utilizado como base para a geração do conteúdo; no grafo em questão, o objeto delimita as fundações do edifício a gerar.

```
{
  "Subject": "ProjectInfo",
  "Data": {
    "Graphs": [
      {
        "name": "House",
        "parameters": [
          {
            "name": "Height",
            "type": "Single"
          },
          {
            "name": "Roof Height",
            "type": "Single"
          }
        ],
        "inputs": [
          {
            "name": "Foundation"
          }
        ]}]}}
```

Figura 4.4: Exemplo de uma mensagem JSON com informação relativa ao projeto Construct a abrir

Entre as duas aplicações, existem diferenças quanto à especificação dos respetivos tipos. Posto isto, é necessário associar as suas equivalências entre os dois sistemas. Na tabela seguinte, é possível verificar os mapeamentos realizados entre os tipos utilizados internamente nos parâmetros de um grafo *Construct* e o respetivo equivalente no ambiente *Blender*. No ambiente *Blender*, os tipos de parâmetros são definidos via funções do módulo *bpy.props*. Estas funções definem o que o *Blender* intitula de *Properties*, que não podem ser utilizadas diretamente, apenas para definir atributos de classes que estendam funcionalidades do *Blender*. Para além de definirem um atributo, podem também definir restrições sobre essas propriedades, tais como os valores mínimo e máximo.

## 4.4 Construção da Interface

A API *Python* do *Blender* está estruturada de forma a que qualquer ferramenta ou operação interna seja definida como uma subclasse do tipo *bpy.types.Operator*: um operador é definido pelas suas propriedades, do tipo *bpy.props*, e os seguintes métodos, que são herdados de *Operator* e podem ser re-definidos por cada nova sub-classe:

**poll** Este método é executado antes de *invoke* e *execute* para verificar se o operador pode ser executado. Caso a validação falhe, o método *execute* do operador não é executado e a interface, definida por *draw*, é bloqueada.

**execute** Corre sempre que o *Operator* é requisitado via a interface do *Blender*, consola *Python* do *Blender* ou em código.

**draw** Utilizado para definir a interface do operador, tipicamente na *tool-bar* do *Blender*, onde os valores das suas propriedades podem ser editados.

Tabela 4.2: Mapeamentos de tipos entre parâmetros Construct e propriedades Blender

| **Construct Type** | **Blender Type** |
|---|---|
| System.Int32 | bpy.props.IntProperty() |
| System.Boolean | bpy.props.BoolProperty() |
| System.Single | bpy.props.FloatProperty() |
| System.Double | bpy.props.FloatProperty() |
| System.String | bpy.props.StringProperty() |
| Construct.Geometry3D.basic.Vector3D | bpy.props.FloatVectorProperty() |

Depois de recebido o objeto *JSON* com a informação do projeto *Construct* a abrir, este é interpretado e é definida uma nova classe que estende a classe base *bpy.types.Operator* por cada grafo do projeto. A cada operador de um grafo definido, são atribuídos, como atributos, os parâmetros do grafo seguindo o mapeamento definido pela tabela 4.2, assim como as entradas de geometria necessárias para a execução do grafo. É definida uma nova secção na barra de ferramentas do *Blender* intitulada de *Construct*, onde serão colocados botões para acionar a requisição destes operadores, que efetuam o pedido da execução dos grafos ao servidor 4.5.

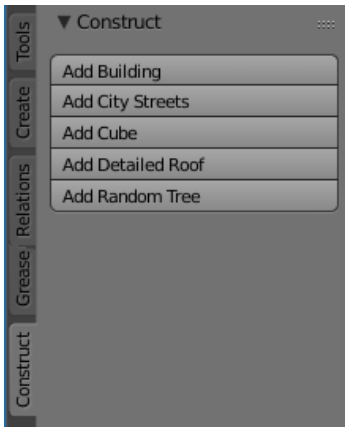


Figura 4.5: Barra de ferramentas depois de aberto um projeto

## 4.5 Execução

Na execução de um operador, os valores dos parâmetros e geometria necessária para execução são compilados num objeto *JSON* 4.6 e enviados ao servidor por meio da ligação por *Sockets*; a geometria resultante é definida como um ficheiro *Wavefront .obj* que é colocado numa pasta temporária partilhada entre as duas aplicações.

```
{
  "Subject": "Generate",
  "Data": {
    "graph_name": "House",
    "project": "..\\proj1.cprj",
    "parameters": [
      {
        "name": "height",
        "value": "5"
      },
      {
        "name": "roof height",
        "value": "5"
      }
    ],
    "inputs": [
      {
        "gate_name": "Base",
        "file_path": "..\\plane.obj"
      }
    ]
  }
}
```

Figura 4.6: *JSON* enviado ao servidor para a execução de um grafo

Na figura 4.7, é possível ver um plano previamente adicionado à cena ser utilizado como *Input* para a execução do grafo *Building*, em conjunto com os valores definidos nos parâmetros: *height*, *roof*, *roof height*, *Door at face*, *Door at position* e *Has windows on lower*; com base nestes valores, é produzido o edifício que pode ser encontrado em 4.8.

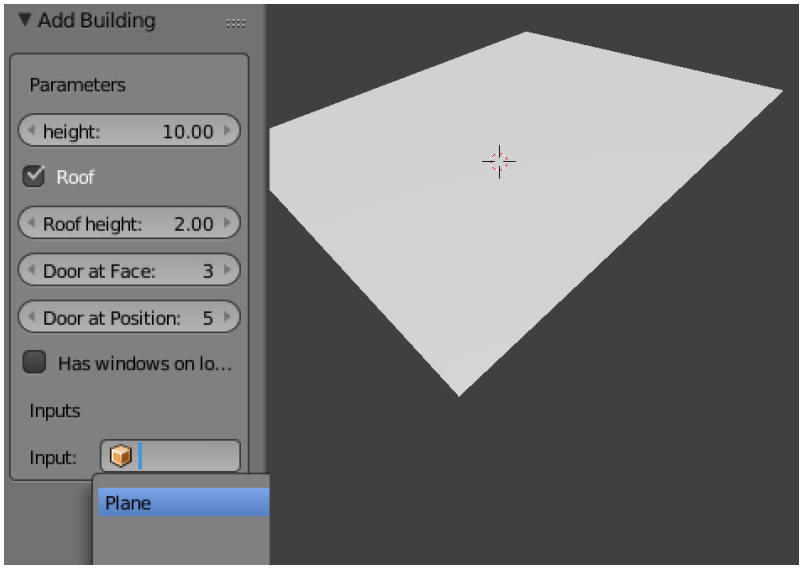


Figura 4.7: Edifício a ser construido e adicionado à cena utilizando *Plane* como base

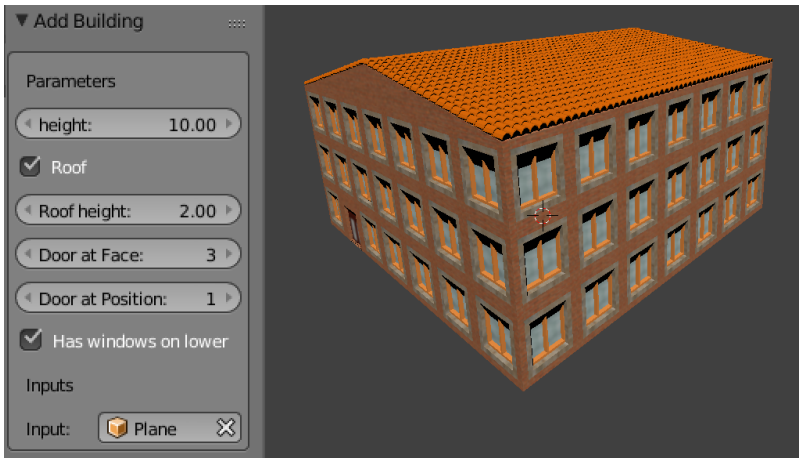


Figura 4.8: Edifício depois de adicionado à cena

A alteração destes valores na interface do operador é refletida em tempo real na cena, como é possível ver em 4.9, onde os valores de *Roof*, *Roof height*, *Door at Position* e *Has windows on lower* foram alterados, produzindo assim um edifício sem um telhado detalhado, com um beiral de apenas 0.3 unidades e a porta colocada numa posição diferente e construído sobre o objeto *plane*.

Na figura 4.10 é possível contemplar que os objetos de entrada podem ser manipulados manualmente antes de estes serem utilizados como base para a construção de um edifício, dando assim

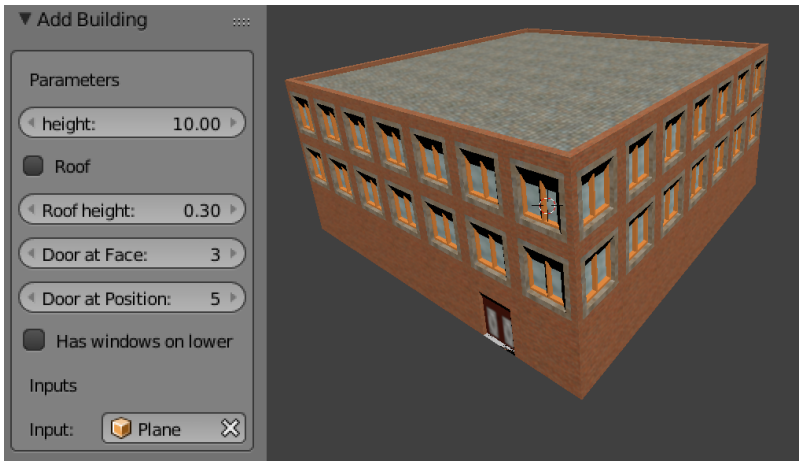


Figura 4.9: Edifício após a alteração de alguns valores

ao modelador uma maior liberdade criativa sobre o conteúdo a ser produzido procedimentalmente.

Com o edifício inserido na cena e a realização de outra operação do *Blender*, e.g. uma translação, a interface de edição dos valores como vista em 4.7 é substituída pela interface da operação utilizada mais recentemente; assim restringindo ao utilizador da alteração dos valores. Trata-se de uma limitação presente em todas as ferramentas do *Blender*, fazendo parte do método de trabalho que este impõe aos seus utilizadores. O programa espera que os utilizadores realizem operações de reversão das alterações (*Ctrl*+*Z*) e voltem a realizar a operação que pretendiam, sem o acesso a um histórico das alterações realizadas.

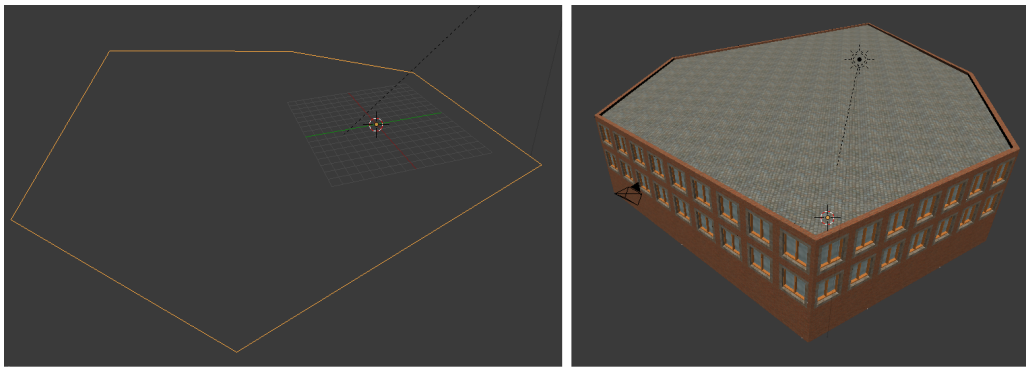

Figura 4.10: Edifício a ser construido sobre um polígono não regular

Numa tentativa de circundar esta limitação, os valores dos parâmetros foram introduzidos a cada objeto como *Custom Properties* (Figura 4.11). Esta abordagem mostrou-se altamente limitada, na medida em que as *Custom Properties* de um dado objeto na cena não podem ser do tipo *bpy.props* e é permitido ao utilizador introduzir valores incompatíveis com o parâmetro subjacente.

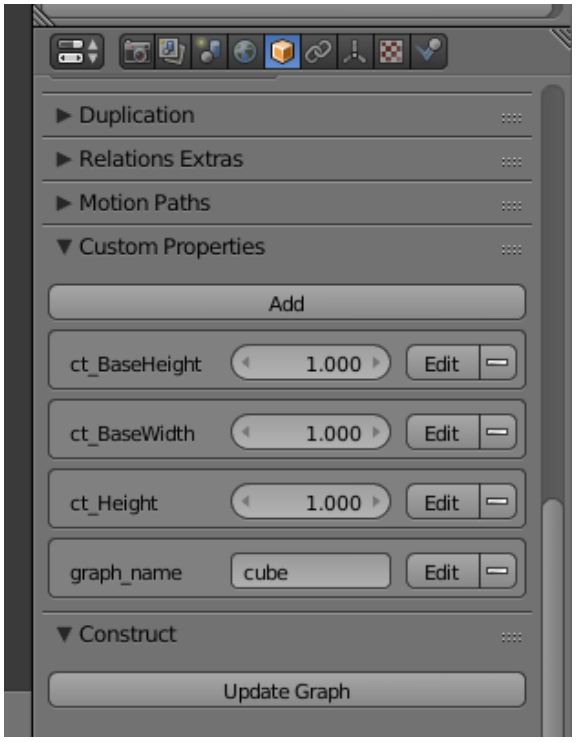


Figura 4.11: *Custom Properties* de um objeto produzido por um grafo *Cube*

# Capítulo 5

# Teste da Solução e Resultados Obtidos

Neste capítulo, será descrita a abordagem utilizada para a realização dos testes com utilizadores da solução detalhada no capítulo anterior. Será abordada a metodologia de teste utilizada, assim como as métricas para a consequente análise dos resultados.

## 5.1 Metodologia de Teste

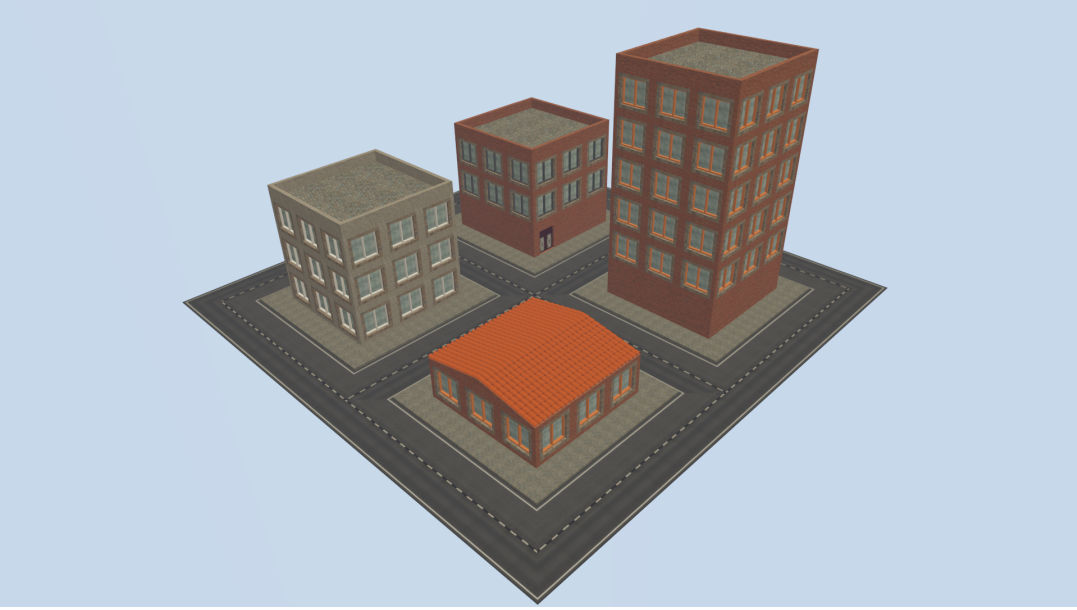

Figura 5.1: Cenário requisitado aos utilizadores

Para realizar o estudo da usabilidade de operações de geração procedimental integradas num ambiente de modelação manual, é necessário avaliar a eficiência com que os modeladores realizam uma tarefa de construção de um cenário. Num ambiente de teste ideal, seria pedido a utilizadores

experientes em *Blender* para construirem um cenário similar ao da figura 5.1, utilizando apenas os recursos nativos à aplicação *Blender*. De seguida, seria pedido para produzirem o mesmo resultado utilizando a solução desenvolvida; desta forma, tornar-se-ia possível comparar dados concretos ente uma abordagem puramente manual e uma abordagem híbrida. Para a realização de um teste deste tipo, seria necessário alocar múltiplas horas para cada utilizador. Devido ao tempo disponível para a realização destes testes ser limitado, optou-se realizar apenas o teste da abordagem desenvolvida, reduzindo o tempo de teste para aproximadamente 30 minutos por utilizador. Para a realização destes testes foi definido um ambiente urbano, figura 5.1, constituído por quatro quarteirões de uma cidade, em que cada lote contém um edifício com características distintas. O teste consiste na recriação, por parte dos sujeitos, do quarteirão no *software Blender* utilizando os recursos disponibilizados por um projeto Construct fornecido. O projeto *Construct* é intitulado de *Urban*, e disponibiliza os grafos necessários para a geração do cenário requisitado, entre os quais se destacam os seguintes:

**Building** é capaz de produzir um edifício sobre uma geometria base, as fachadas deste edifício são divididas numa grelha composta por janelas *Window*, as janelas ao nível do solo são opcionais; a posição da porta do edifício, configurável; e um telhado do tipo *DetailedRoof*, opcional. Grafo em anexo B.1.

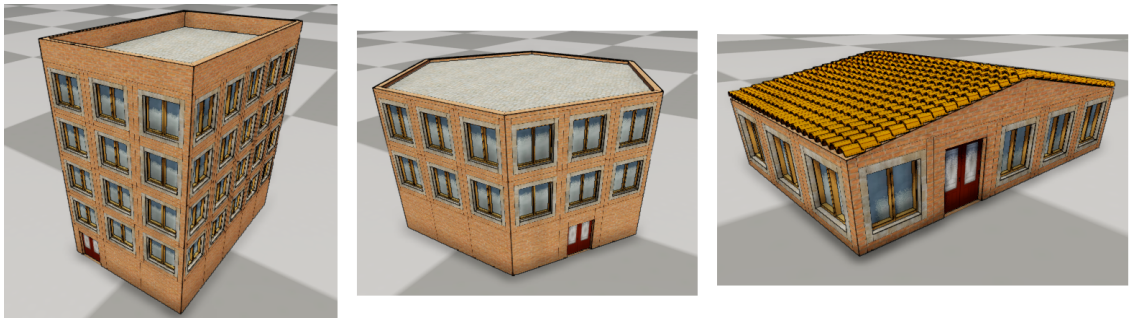

Figura 5.2: Exemplos de resultados possíveis do grafo *Building*

**Window** Este grafo resulta numa janela com um rebordo. Os limites deste grafo são definidos por uma geometria base, por exemplo uma célula da fachada do edifício produzido por *Building*, grafo em anexo B.3.

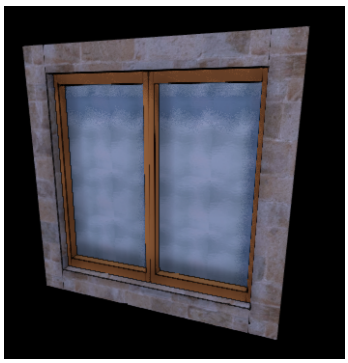

Figura 5.3: Exemplo de resultado do grafo *Window*

**Door** Este grafo resulta numa porta com um rebordo. A porta é construída sobre uma superfície definida por uma geometria que lhe é passada como entrada. No grafo *Building* este é aplicado a uma das células do primeiro nível de uma fachada, grafo em anexo B.5.

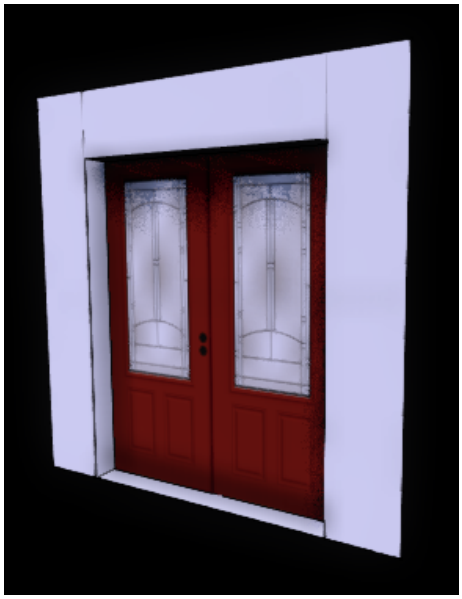

Figura 5.4: Exemplo de resultado do grafo *Door*

**CityStreets** Através de ferramentas nativas ao *Construct* definidas no grafo *RectangularGrid*, é possível a a geração de um sistema de ruas e quarteirões de uma cidade. Como parâmetros de *CityStreets*, a área da cidade e o tamanho de cada lote podem ser delineados, grafo em anexo B.2.

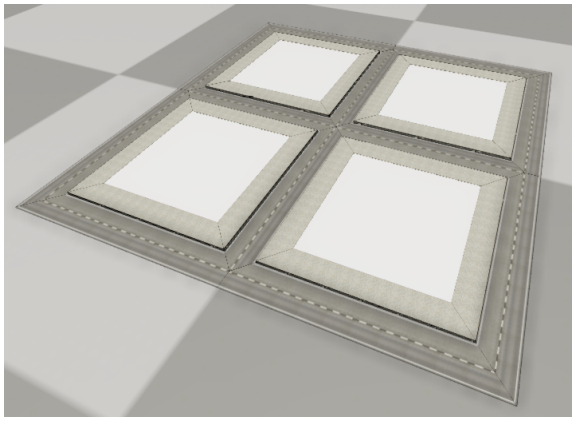

Figura 5.5: Exemplo de resultado do grafo *CityStreets*

Depois de estabelecidos estes recursos de geração procedimental para serem utilizados no decorrer dos testes, a execução de cada teste com cada utilizador estava dividido nos seguintes passos:

1. Apresentação do projeto, contexto, motivação e solução desenvolvida;
2. Resposta a um questionário inicial para avaliar o nível de experiência com a ferramenta *Blender*, apresentar o cenário da figura 5.1 e questionar quanto tempo seria necessário para produzir um cenário equivalente, por meios puramente manuais;
3. Demonstração do funcionamento da solução, dando alguns exemplos de utilização;
4. Execução do teste por parte do utilizador, o tempo era cronometrado;
5. Resposta a um segundo questionário com perguntas standardizadas para a realização de um teste de usabilidade;
6. Discussão com o utilizador.

Os testes com utilizadores foram avaliados nos seguintes três parâmetros: Tempo estimado pelos utilizadores para produzirem um cenário similar ao da figura 5.1, utilizando apenas funcionalidades nativas ao Blender; tempo utilizado a construir um cenário similar ao da figura 5.1 utilizando o *add-on* desenvolvido, recorrendo aos grafos *Construct* previamente especificados. Finalmente, foi requisitada a resposta a um questionário no final do teste, em que algumas das questões eram relativas à usabilidade do sistema.

Para este questionário final, foi realizado um teste padrão criado por Brooke J. [Bro96] intitulado de *System Usability Scale*. O teste permite a avaliação da usabilidade de um sistema através das respostas obtidas às suas dez perguntas padronizadas. A partir destes resultados, é possível obter uma pontuação de zero a cem de modo a comparar a usabilidade do sistema com a de outras aplicações. O teste consiste nas seguintes dez questões padrão (no contexto deste teste a palavra "sistema"foi substituída por "add-on"):

1. Gostaria de usar esta add-on com frequência
2. O add-on é desnecessariamente complexo
3. O add-on é fácil de utilizar
4. Preciso de ajuda para operar o add-on
5. As diversas funções deste sistema foram bem integradas
6. Existem muitas inconsistências no add-on
7. Muitas pessoas iriam utilizar este add-on rapidamente
8. O add-on é muito complicado de utilizar
9. Eu senti-me muito confiante com o add-on
10. É preciso aprender muitas coisas antes de utilizar o add-on

Estas são respondidas numa escala de *Likert* [Lik32], que especifica o nível de concordância com a afirmação colocada dividido numa escala de valores de 1 a 5, nesta o valor de 1 corresponde a *Discordo Totalmente* e 5 corresponde a *Concordo Totalmente*.

A pontuação é obtida realizando um somatório dos valores atribuídos a cada questão, seguindo duas regras:

1. Aos valores das perguntas ímpares é subtraído uma unidade.
2. 5 é subtraído a cada valor de uma pergunta par.

O resultado deste somatório é multiplicado por 2.5, obtendo assim a pontuação de usabilidade do sistema. O valor de 68 pontos é considerado a média, como tal um sistema com uma pontuação superior a 68 é considerado positivo. [Bro13]

## 5.2 Resultados

Os testes foram realizados com utilizadores com alguma experiência de utilização da ferramenta *Blender*. Foram convidados alunos da Faculdade de Belas Artes da Universidade do Porto com alguns conhecimentos de modelação e também com alguma experiência de modelação manual. Das sete pessoas inicialmente inscritas no evento apenas quatro acabaram por ter disponibilidade para realizar o teste, sendo três destes alunos da FBAUP e o outro externo ao ambiente universitário.

No geral, o *feedback* obtido da solução foi positivo. Nenhum dos utilizadores tinha utilizado previamente uma ferramenta para a geração automática de conteúdo, e dois dos utilizadores não sabiam da existência de métodos de geração procedimental de conteúdos tridimensionais para a construção de cenários, apesar de conhecerem exemplos de vídeo jogos com ambientes gerados procedimentalmente. Durante os testes, os utilizadores não revelaram dificuldades significativas em aprender o funcionamento da ferramenta. Para eles, a ferramenta estava bem integrada na interface do *Blender* e no *workflow* a que já estão acostumados.

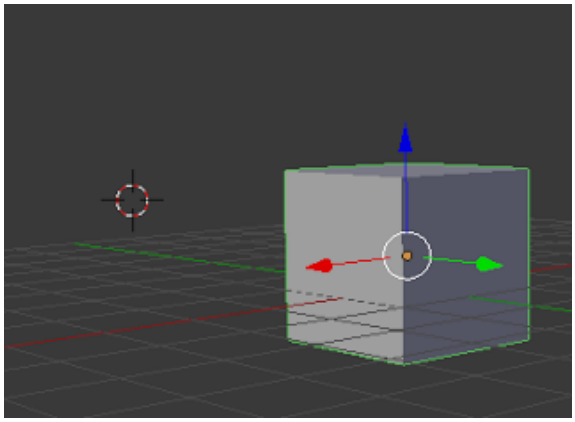

Figura 5.6: Cursor 3D do Blender

Um dos pontos que causou alguma frustração para os utilizadores na utilização da solução estava relacionado com a interação do *add-on* com o Cursor 3D do *Blender* 5.6. Numa interação normal com o *Blender*, os utilizadores mudam a posição do cursor constantemente, e numa situação em que estes requisitassem a geração de um novo conteúdo, o resultado iria sempre ser posicionado sobre o cursor 3D; desta forma, em situações em que o posicionamento era esperado sobre uma geometria base, a interação tornava-se menos intuitiva. No entanto, este é um paradigma comum a várias ferramentas nativas do *Blender*, sendo o seu resultado sempre colocado na posição do cursor 3D.

No caso do utilizador 3, o mais lento a realizar o teste, o na tabela 5.1, é possível observar a relativa inexperiência deste com a ferramenta *Blender*; simultaneamente reconhecida pelo próprio e também suportada pela respetiva estimativa ingénua para o tempo de modelação manual: em comparação com o utilizador mais experiente, o utilizador 1, que completou o teste em apenas

Tabela 5.1: Comparação entre o tempo estimado a modelar o cenário 5.1 manualmente e o tempo utilizado para construir o cenário utilizando o *add-on* desenvolvido

| **Utilizador** | **Nível de exp. com *Blender*** | **Freq. de utilização do Blender** | **Estimativa para a MM** | **Utilizando o *Add-on*** |
|---|---|---|---|---|
| *1* | "Avançado" | Diariamente | 1 <>3h | 16min |
| *2* | "Intermédio" | Algumas vezes por mês | +3h | 19min |
| *3* | "Intermédio" | Algumas vezes por mês | +3h | 26min |
| *4* | "Novato" | Diariamente | <1h | 20min |

16 minutos e demonstrou uma rápida adaptação à solução desenvolvida. Tendo em conta que a solução desenvolvida requer conhecimentos prévios com a ferramenta *Blender*, seria de esperar algumas dificuldades na realização do teste por parte dos utilizadores menos experientes. No entanto, o tempo utilizado para produzir o resultado esperado por parte de utilizadores menos experientes continua a ser um bom resultado comparativamente à abordagem manual, com a qual estavam familiarizados. Como é possível constatar na tabela 5.1, o tempo necessário para construir o cenário utilizando o *add-on* superou as expetativas dos utilizadores. Estes ficaram surpreendidos com a facilidade com que era possível adicionar conteúdo gerado procedimentalmente, realizar alterações aos parâmetros e utilizá-los num ambiente de modelação manual, para rapidamente construírem uma base de trabalho que pode facilmente ser alterada manualmente e resultar no cenário desejado.

Tabela 5.2: Resultados da avaliação de usabilidade do sistema

| **Utilizador** | **Q1** | **Q2** | **Q3** | **Q4** | **Q5** | **Q6** | **Q7** | **Q8** | **Q9** | **Q10** | **Pontuação** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *1* | 4 | 2 | 3 | 3 | 4 | 3 | 5 | 3 | 4 | 4 | 62.5 |
| *2* | 4 | 2 | 4 | 4 | 4 | 1 | 4 | 1 | 4 | 2 | 75.0 |
| *3* | 3 | 1 | 4 | 4 | 4 | 2 | 5 | 1 | 3 | 2 | 72.5 |
| *4* | 4 | 2 | 4 | 3 | 4 | 1 | 4 | 2 | 3 | 3 | 70.0 |
| | | | | | | | | | | **Média** | **70.0** |

A avaliação da usabilidade 5.2 resultou numa pontuação total de 70 pontos, sendo assim considerada, um resultado acima da média segundo os *standards* referidos previamente.

No final de cada teste, foi realizada uma pequena discussão com os utilizadores, nesta foi demonstrada satisfação com a agilidade com que produziram e editavam o conteúdo na cena. Também foi constatado um elevado interesse na utilização uma solução similar para agilizarem o processo de modelação nos seus trabalhos e projetos. Foram guardadas duas das cenas resultadas do teste com os utilizadores 3 e 4. Estas são respetivamente o anexo C.1 e o anexo C.2. Nestes é possível observar a proximidade dos resultados com a cena requisitada. No entanto existem algumas pequenas discrepâncias no que toca a texturas e orientações de alguns dos edifícios.

Os resultados obtidos pelo teste de usabilidade coincidem com o *feedback* positivo obtido dos utilizadores envolvidos nos testes.

# Capítulo 6

# Conclusões e Trabalho Futuro

Foi desenvolvida com sucesso uma solução com o intuito de aproximar duas abordagens distintas à modelação. Foi definido uma metodologia com o objetivo de interligar duas aplicações, uma de geração procedimental, a segunda de modelação manual. Através da implementação desta metodologia foi possível introduzir recursos da ferramenta *Construct* no ambiente de modelação *Blender*, foi possível criar um ambiente de modelação híbrido onde os utilizadores podem facilmente adicionar, editar e manipular manualmente conteúdo gerado procedimentalmente.

A solução foi testada com utilizadores experientes em modelação com a ferramenta *Blender* e sem conhecimentos prévios de geração procedimental. O *feedback* dos utilizadores e a consequente avaliação de usabilidade foram positivos, tendo estes demonstrado interesse na existência de uma combinação de uma ferramenta de geração procedimental como o *Construct* com a ferramenta *Blender*. A introdução de recursos de geração procedimental num ambiente de modelação manual é altamente favorável para a agilização da produção de novo conteúdo numa ferramenta de modelação manual, permitindo combinar e aproveitar eficientemente as vantagens das duas vertentes.

## 6.1 Trabalho Futuro

Existem, no entanto, áreas onde podem ser introduzidas melhorias. O trabalho futuro poderá focar-se nos seguintes pontos:

**Servidor embebido na ferramenta** O servidor desenvolvido estava assente numa aplicação dedicada, separando-a assim da ferramenta de edição Construct. Por razões de conveniência, seria desejável que estes recursos estivessem disponíveis diretamente a partir da ferramenta de edição.

**Introdução de mais estruturas de dados** Permitir parametrizar outros tipos de dados, tais como linhas, texturas, animações, luzes, etc. A introdução destes novos tipos de estruturas iria aumentar a flexibilidade na utilização da solução desenvolvida.

**Otimizar a comunicação** Devido ao foco desta dissertação não ser a comunicação entre duas aplicações, esta foi implementada para apenas satisfazer as necessidades básicas da solução. Uma solução multi-cliente poderia ser explorada. Permitindo assim a um grupo de modeladores partilharem recursos de geração procedimental.

**Interação via nós no *Blender*** Implementar e testar uma interação por paradigma de interação por grafos, pode potencialmente facilitar a manipulação dos valores dos parâmetros de um conteúdo a ser adicionado à cena. Refletindo o paradigma utilizado no *Construct*, o conceito é atualmente utilizado por modeladores em *Blender* para a edição de materiais.

# Referências


[ARB07] Daniel G. Aliaga, Paul A. Rosen e Daniel R. Bekins. Style grammars for interactive visualization of architecture. *IEEE Transactions on Visualization and Computer Graphics*, 13:786–797, 2007. doi:10.1109/TVCG.2007.1024.

[Aut15a] Autodesk. Autodesk 3DS Max, 2015. [Online; accessed 14-Fev-2015]. URL: http://www.autodesk.com/products/3ds-max/overview.

[Aut15b] Autodesk. Autodesk Maya, 2015. [Online; accessed 14-Fev-2015]. URL: http://www.autodesk.com/products/maya/overview.

[Aut15c] Autodesk. Compare to other products, 2015. [Online; accessed 14-Fev-2015]. URL: http://www.autodesk.com/suites/entertainment-creation-suite/compare/compare-to-other-products.

[Bap14] Anderson Baptista. Blender Encyclopedia: Modifiers, 2014. [Online; accessed 14-Fev-2015]. URL: http://www.blenderguru.com/articles/blender-101-modifier-encyclopedia/.

[Bau72] Bruce G. Baumgart. WINGED EDGE POLYHEDRON REPRESENTATION. *National Technical Information Service U. S. DEPARTMENT OF COMMERCE 5285*, (October), 1972. URL: http://www.dtic.mil/dtic/tr/fulltext/u2/755141.pdf.

[Bez68] Pierre E Bezier. How Renault Uses Numerical Control for Car Body Design and Tooling. *SAE International*, 1968. URL: http://www.sae.org/technical/papers/680010, doi:10.4271/680010.

[Béz81] Pierre E. Bézier. *A view of CAD/CAM*, volume 13. 1981. URL: http://linkinghub.elsevier.com/retrieve/pii/0010448581901421, doi:10.1016/0010-4485(81)90142-1.

[ble15a] Blender 3D, 2015. [Online; accessed 16-Fev-2015]. URL: http://www.blender.org/.

[ble15b] Blender Extensions, 2015. [Online; accessed 16-Fev-2015]. URL: http://wiki.blender.org/index.php/Doc:2.6/Manual/Extensions.

[ble15c] Blender Scripts Development, 2015. [Online; accessed 16-Fev-2015]. URL: http://wiki.blender.org/index.php/Dev:2.5/Py/Scripts.

[Ble15d] Input Nodes — Blender Reference Manual, 2015. [Online; accessed 14-Fev-2015]. URL: https://www.blender.org/manual/render/blender{_}render/materials/nodes/types/input.html.

[Ble15e] Modifiers and deformation, 2015. [Online; accessed 14-Fev-2015]. URL: http://wiki.blender.org/index.php/Doc:2.4/Manual/Modifiers.

[Bro96] John Brooke. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[Bro13] John Brooke. SUS: A Retrospective. 8(2):29–40, 2013. URL: http://www.usabilityprofessionals.org.

[BWS10] Martin Bokeloh, Michael Wand e Hans-Peter Seidel. A connection between partial symmetry and inverse procedural modeling. *ACM Transactions on Graphics*, 29(4):1, 2010. doi:10.1145/1833351.1778841.

[Bé83] Pierre E. Bézier. *UNISURF, from styling to tool-shop*, volume 4. 1983. doi:10.1016/0166-3615(83)90017-9.

[Cab92] J.J.S.P. Cabral. An introduction to splines for use in computer graphics & geometric modeling. 9:278, 1992. doi:10.1016/0955-7997(92)90106-H.

[Cob84] Elizabeth Susan Cobb. *Design of Sculptured Surfaces Using the B-spline Representation*. PhD thesis, 1984. AAI8420751.

[Cod15] Catlike Coding. Numberflow, procedural texture editor for Unity. http://catlikecoding.com/numberflow/, 2015. [Online; accessed 14-Fev-2015].

[dCB09] Giliam J. P. de Carpentier e Rafael Bidarra. Interactive GPU-based procedural heightfield brushes. *Proceedings of the 4th International Conference on Foundations of Digital Games - FDG '09*, page 55, 2009. URL: http://portal.acm.org/citation.cfm?doid=1536513.1536532, doi:10.1145/1536513.1536532.

[FPRJ00] Sarah F Frisken, Ronald N Perry, Alyn P Rockwood e Thouis R Jones. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 249–254. ACM Press/Addison-Wesley Publishing Co., 2000.

[GMSe09] James Gain, Patrick Marais e Wolfgang Straß er. Terrain sketching. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games - I3D '09*, page 31, 2009. URL: http://dl.acm.org/citation.cfm?id=1507149.1507155, doi:10.1145/1507149.1507155.

[Hol15] Joachim Holmér. Shader Forge. http://acegikmo.com/shaderforge/, 2015. [Online; accessed 14-Fev-2015].

[IOI06a] Takashi Ijiri, Shigeru Owada e Takeo Igarashi. Seamless integration of initial sketching and subsequent detail editing in flower modeling. *Computer Graphics Forum*, 25:617–624, 2006. doi:10.1111/j.1467-8659.2006.00981.x.

[IOI06b] Takashi Ijiri, Shigeru Owada e Takeo Igarashi. The Sketch L-System: Global Control of Tree Modeling Using Free-Form Strokes. In *International Symposium on Smart Graphics*, pages 138 – 146, 2006. doi:10.1007/11795018_13.

[Jon02] Jeff Jonaitis. Box Modeling Technique, 2002. URL: http://www.jjonaitis.com/tuto/tuto2.htm.

[KK12] Lars Krecklau e Leif Kobbelt. Interactive modeling by procedural high-level primitives. *Computers & Graphics*, 36(5):376–386, 2012. URL: http://www.sciencedirect.com/science/article/pii/S0097849312000672.

[KR79] Joseph Kung e Gian-Carlo Rota. A practical guide to splines. 32:81, 1979. doi:10.1016/0001-8708(79)90033-1.

[Lik32] Rensis Likert. A technique for the measurement of attitudes. 1932.

[Lin68] A Lindenmayer. Mathematical models for cellular interaction in development, Parts I and II. *Journal of Theoretical Biology*, 18:280–315, 1968.

[LRBP12] Steven Longay, Adam Runions, Frédéric Boudon e Przemyslaw Prusinkiewicz. TreeSketch : Interactive Procedural Modeling of Trees on a Tablet. *The proceedings of the Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pages 107–120, 2012. doi:10.2312/SBM/SBM12/107-120.

[LWW08] Markus Lipp, Peter Wonka e Michael Wimmer. *Interactive visual editing of grammars for procedural architecture*, volume 27. 2008. doi:10.1145/1360612.1360701.

[MGP06] Niloy J. Mitra, Leonidas J. Guibas e Mark Pauly. Partial and approximate symmetry detection for 3D geometry. *ACM SIGGRAPH 2006 Papers on - SIGGRAPH '06*, page 560, 2006. URL: http://dl.acm.org/citation.cfm?id=1141924$\delimiter"026E30F$nhttp://portal.acm.org/citation.cfm?doid=1179352.1141924, doi:10.1145/1179352.1141924.

[MP96] Radomír Měch e Przemyslaw Prusinkiewicz. Visual models of plants interacting with their environment. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 96:397–410, 1996. URL: http://dl.acm.org/citation.cfm?id=237279, doi:10.1145/237170.237279.

[MWH+06] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer e Luc Van Gool. *Procedural modeling of buildings*, volume 25. 2006. doi:10.1145/1141911.1141931.

[Pat12] Gustavo Patow. User-friendly graph editing for procedural modeling of buildings. *IEEE Computer Graphics and Applications*, 32(2):66–75, 2012. doi:10.1109/MCG.2010.104.

[PF01] Ronald N Perry e Sarah F Frisken. Kizamu: A system for sculpting digital characters. *Computer Graphics Annual Conference (SIGGRAPH 2001)*, pages 47–56, 2001. URL: http://cdc310-www.scopus.com/inward/record.url?eid=2-s2.0-0035152623&partnerID=40&md5=13c79efac9e9afa8d78dae0157a9bbb9, doi:10.1145/383259.383264.

[PHL+09] Wojciech Palubicki, Kipp Horel, Steven Longay, Adam Runions, Brendan Lane, Radomír Měch e Przemyslaw Prusinkiewicz. Self-organizing tree models for image synthesis. 28:1, 2009. doi:10.1145/1531326.1531364.

[Pix15] Pixologic. Zbrush Overview, 2015. [Online; accessed 14-Fev-2015]. URL: http://pixologic.com/zbrush/features/overview/.

[PLH88] Przemyslaw Prusinkiewicz, Aristid Lindenmayer e James Hanan. Development models of herbaceous plants for computer imagery purposes. 22:141–150, 1988. doi:10.1145/378456.378503.

[Pru00] Przemyslaw Prusinkiewicz. Simulation modeling of plants and plant ecosystems. 43:84–93, 2000. doi:10.1145/341852.341867.

[ŠBM+10] Ondrej Šťava, Bedrich Beneš, Radomir Měch, Daniel G Aliaga e Peter Krištof. Inverse procedural modeling by automatic generation of L-systems. *Computer Graphics Forum*, 29(2):665–674, 2010. doi:10.1111/j.1467-8659.2009.01636.x.

[SM08] Karan Singh e James Mccrae. Sketch-Based Path Design. pages 95–102, 2008.

[SMBC13] Pedro Brandão Silva, Pascal Müller, Rafael Bidarra e António Coelho. Node-based shape grammar representation and editing. *PCG*, pages 1–8, 2013. URL: https://graphics.tudelft.nl/Publications-new/2013/SMBC13a/SMBC13a.pdf.

[Smi06] Colin Smith. On Vertex-Vertex Systems and Their Use in Geometric and Biological Modelling. *Computer Science*, page 216, 2006. URL: http://algorithmicbotany.org/papers/smithco.dis2006.pdf.

[Spe11] Scott Spencer. *ZBrush Character Creation: Advanced Digital Sculpting*. John Wiley & Sons, 2011.

[STdKB08] Ruben Michaël Smelik, Tim Tutenel, Klaas Jan de Kraker e Rafael Bidarra. A Proposal for a Procedural Terrain Modelling Framework. *EGVE Symposium*, pages 1–4, 2008. URL: http://graphics.tudelft.nl/~rafa/myPapers/bidarra.EGVE08.pdf.

[STdKB10] Ruben Michaël Smelik, Tim Tutenel, Klaas Jan de Kraker e Rafael Bidarra. Interactive creation of virtual worlds using procedural sketching. *Proceedings of . . .*, pages 1–4, 2010. URL: http://graphics.tudelft.nl/Publications-new/2010/STDB10e/STDB10e.pdf.

[sub14] Subdivision modelling, 2014. URL: http://theorangeduck.com/page/subdivision-modelling.

[Tec15] Unity Technologies. Mecanim Animation System. http://docs.unity3d.com/Manual/MecanimAnimationSystem.html, 2015. [Online; accessed 14-Fev-2015].

[TM06] Robert F. Tobler e Stefan Maierhofer. A mesh data structure for rendering and subdivision. *Proceedings of WSCG (International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision)*, pages 157–162, 2006. URL: http://visplore.net/publications/pdfs/VRVis_2005_11_08_08_27_27.pdf.

[VGDA+12] Carlos a. Vanegas, Ignacio Garcia-Dorado, Daniel G. Aliaga, Bedrich Benes e Paul Waddell. Inverse design of urban procedural models. *ACM Transactions on Graphics*, 31:1, 2012. URL: http://dl.acm.org/citation.cfm?doid=2366145.2366187, doi:10.1145/2366145.2366187.

[WP95] Jason Weber e Joseph Penn. Creation and rendering of realistic trees. *SIGGRAPH '95 - Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 119–128, 1995. doi:10.1145/218380.218427.

[WWSR03] Peter Wonka, Michael Wimmer, François Sillion e William Ribarsky. *Instant architecture*, volume 22. 2003. doi:10.1145/882262.882324.

# REFERÊNCIAS

# Anexo A

# Mensagens JSON

## A.1 *JSON* de Pedido de Abertura do Projeto *Urban*

```
{
  "Subject": "OpenProj",
  "Data": "..\\Urban.cprj"
}
```

## A.2 *JSON* de Informação do Projeto *Urban*

```
{
  "Subject": "ProjectInfo",
  "Data": {
   {
  "Graphs": [
    {
      "name": "Building",
      "parameters": [
        {
          "name": "height",
          "type": "Single"
        },
        {
          "name": "Roof",
          "type": "Boolean"
        },
        {
          "name": "Roof height",
          "type": "Single"
```

```
      },
      {
        "name": "Door at Face",
        "type": "Int32"
      },
      {
        "name": "Door at Position",
        "type": "Int32"
      },
      {
        "name": "Has windows on lower",
        "type": "Boolean"
      }
    ],
    "inputs": [
      {
        "name": "Input"
      }
    ]
  },
  {
    "name": "City Streets",
    "parameters": [],
    "inputs": []
  },
  {
    "name": "Cube",
    "parameters": [
      {
        "name": "Base Width",
        "type": "Single"
      },
      {
        "name": "Base Height",
        "type": "Single"
      },
      {
        "name": "Height",
        "type": "Single"
      }
```

```
      ],
      "inputs": []
    },
    {
      "name": "Detailed Roof",
      "parameters": [],
      "inputs": [
        {
          "name": "Input/3/0"
        }
      ]
    }
  ]
}
  }
}
```

## A.3 Exemplo de Pedido para Gerar um Grafo

```
{
  "Subject": "Generate",
  "Data": {
    "graph_name": "House",
    "project": "..\\test.cprj",
    "parameters": [
      {
        "name": "height",
        "value": "5"
      },
      {
        "name": "roof height",
        "value": "5"
      }
    ],
    "inputs": [
      {
        "gate_name": "Base",
        "file_path": "..\\plane.obj"
      }
    ]
```

```
  }
}
```

## A.4 Exemplo de Resposta de Erro do Servidor

```
{
  "Subject": "Error",
  "Data": "Error executing the graph"
}
```

# Anexo B

# Principais Grafos do Projeto *Urban*

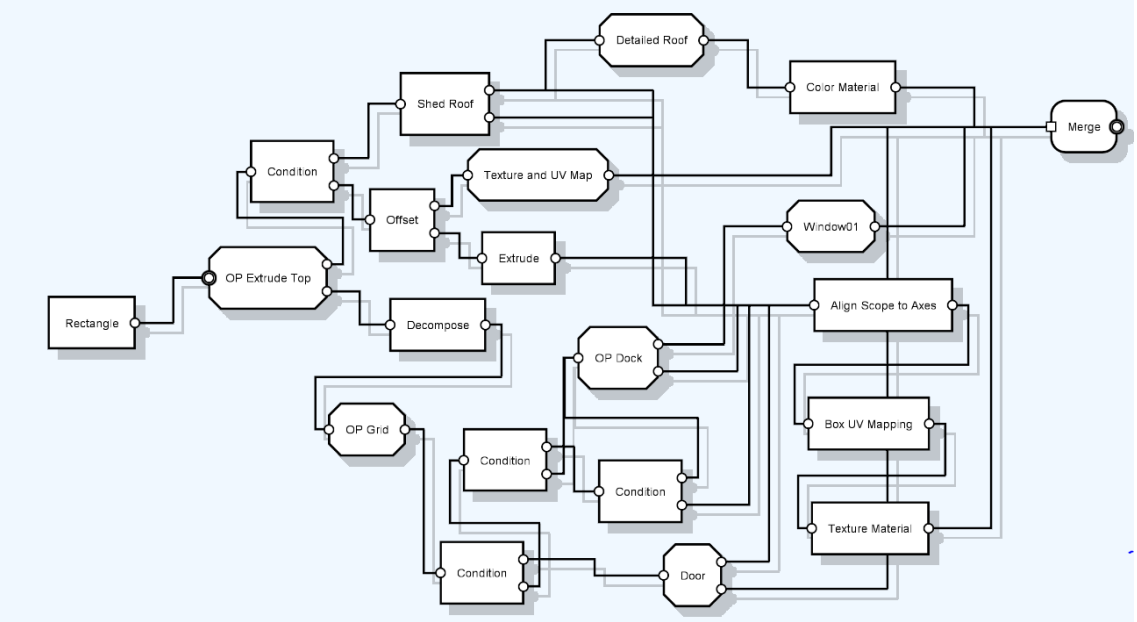


Figura B.1: Grafo *Building* como visto no editor *Construct*

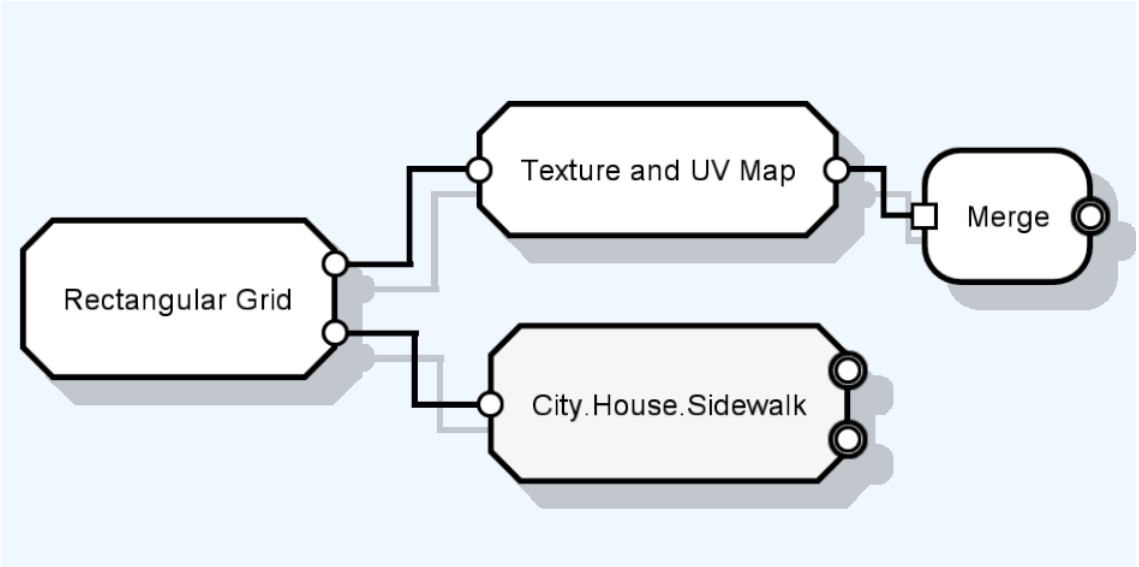


Figura B.2: Grafo *CityStreets* como visto no editor *Construct*

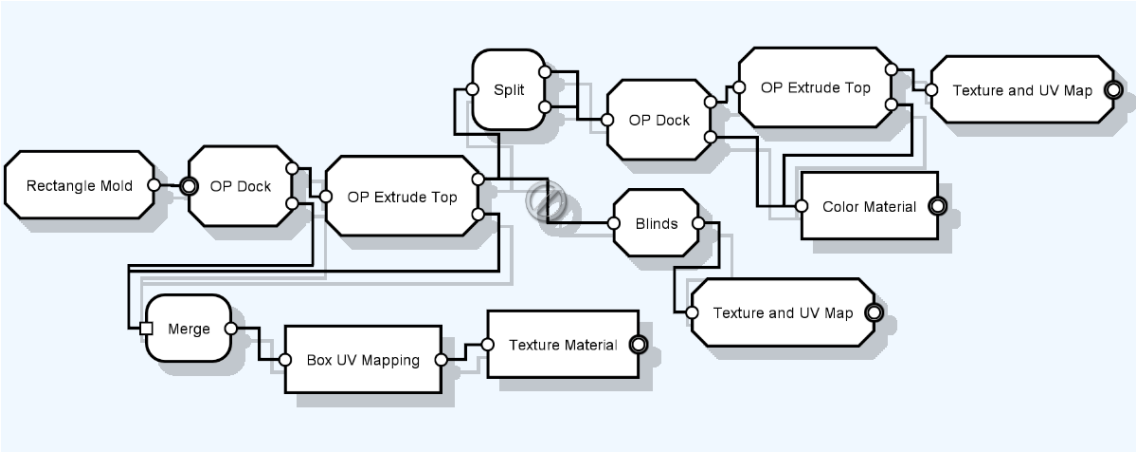


Figura B.3: Grafo *Window* como visto no editor *Construct*

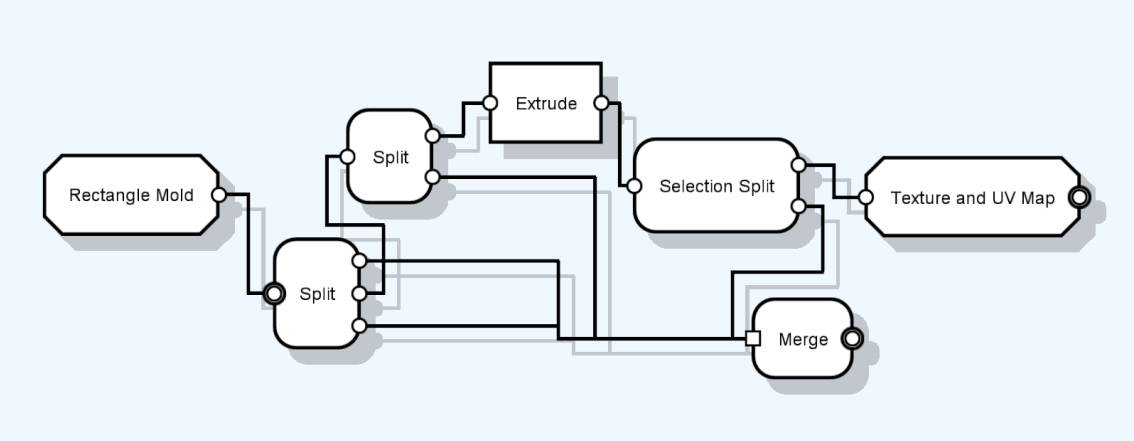


Figura B.4: Grafo *Door* como visto no editor *Construct*

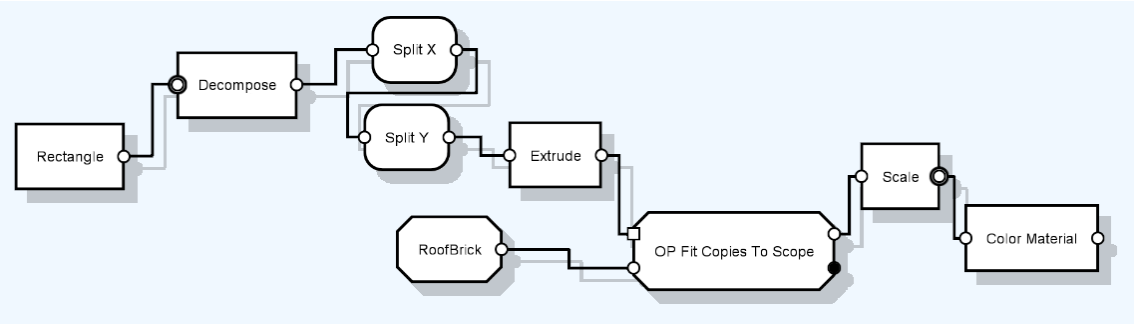


Figura B.5: Grafo *DetailedRoof* como visto no editor *Construct*

# Anexo C

# Cenários Resultantes dos Testes com Utilizadores

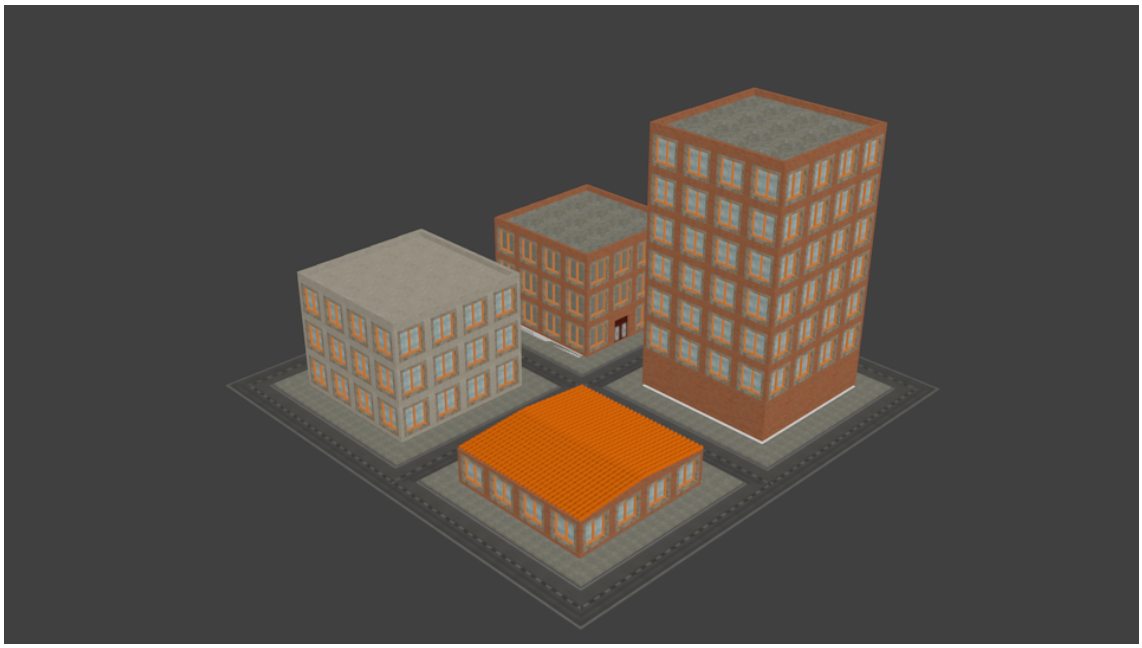

Figura C.1: Cenário criado pelo utilizador 3

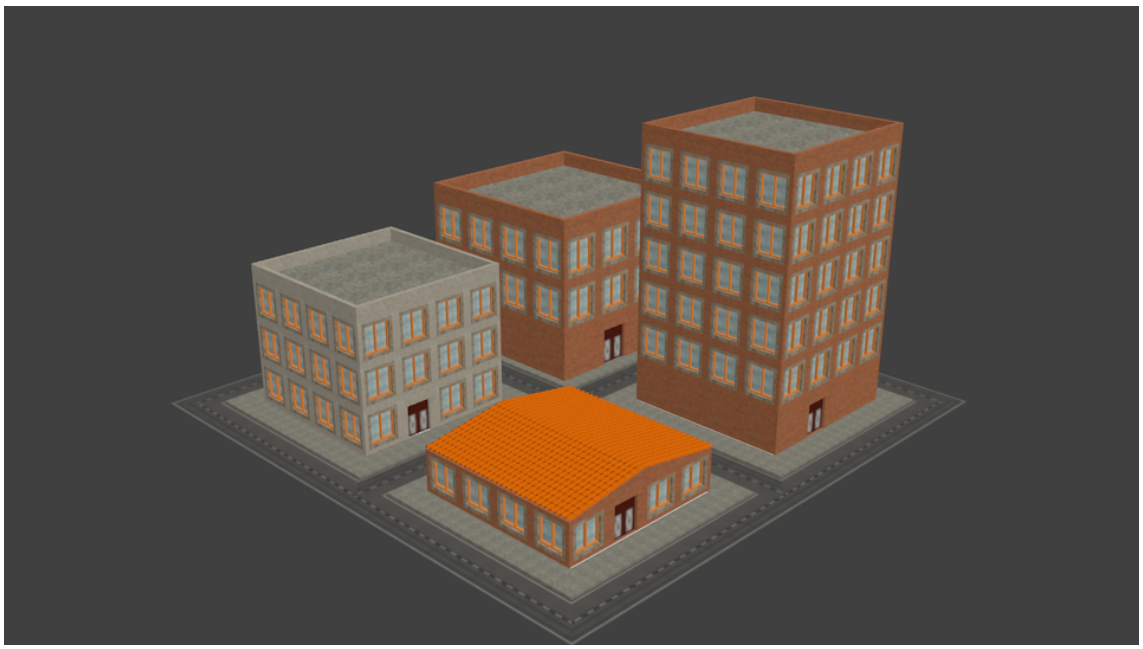

Figura C.2: Cenário criado pelo utilizador 4